

k-Position, Follow, Equation, k-C-Continuation Tree Automata Constructions

Ludovic Mignot, Nadia Ouali Sebti and Djelloul Ziadi *

Laboratoire LITIS - EA 4108 Université de Rouen, Avenue de l'Université
76801 Saint-Étienne-du-Rouvray Cedex.
{ludovic.mignot, nadia.ouali-sebti, djelloul.ziadi}@univ-rouen.fr

Abstract. There exist several methods of computing an automaton recognizing the language denoted by a given regular expression: In the case of words, ones can compute the *position automaton* \mathcal{P} due to Glushkov, the *c-continuation automaton* \mathcal{C} due to Champarnaud and Ziadi, the *follow automaton* \mathcal{F} due to Illie and Yu and the equation automaton \mathcal{E} due to Antimirov. It has been shown that \mathcal{P} and \mathcal{C} are isomorphic and that \mathcal{E} (resp. \mathcal{F}) is a quotient of \mathcal{C} (resp. of \mathcal{P}).

In this paper, we define from a given regular tree expression the *k-position tree automaton* \mathcal{P} and the *follow tree automaton* \mathcal{F} . Using the definition of the equation tree automaton \mathcal{E} of Kuske and Meinecke and our previously defined *k-C-continuation tree automaton* \mathcal{C} , we show that the previous morphic relations are still valid as far as tree expressions are concerned.

1 Introduction

Regular expressions are used in numerous domains of applications in computer science. They are an easy and compact way to represent potentially infinite regular languages, that are well-studied objects leading to efficient decision problems. Among them, the membership test, that is to determine whether or not a given word belongs to a language. Given a regular expression E with n symbols and a word w , to determine whether w is in the language denoted by E can be polynomially performed (with respect to n) *via* the computation of a finite state machine, called an automaton, that can be seen as a symbol-labelled graph with initial and final states. Such an automaton can be computed *via* several methods.

The first approach is to determine particular properties over the syntactic structure of the regular expression E . Glushkov [8] proposed the computation of four position functions Null, First, Last, and Follow, which once computed, lead to the computation of a $(n + 1)$ -states automaton. Illie and Yu showed in [9] how to reduce it by merging similar states. Another method is to compute the transition function of the automaton as follows: assimilating a state s to an expression, any path labelled by a word w brings the automaton from a state s into a finite set of states $S' = \{s'_1, \dots, s'_k\}$ such that these states denote the quotient $w^{-1}(L(s))$ of the language $L(s)$ by w , that contains the word w' such that ww' belongs to $L(s)$. Basically, it is a computation that tries to determine what words w' can be accepted after reading a prefix w . The first author that introduced such a process is Brzozowski [2]. He showed how to compute a regular expression denoting $w^{-1}(L(E))$ from the expression E : this expression, denoted by $d_w(E)$, is called the *derivative* of E with respect to w .

Furthermore, the set of dissimilar derivatives, combined with reduction according to associativity, commutativity and idempotence of the sum, is finite and can lead to the computation of a deterministic finite automaton. Antimirov [1] extended this method to the computation of partial derivatives, that are no longer expressions but sets of expressions. These so-called derived terms produce the *equation automaton*. Finally, by deriving expressions after having them indexed, Champarnaud and Ziadi [4] computed the *c-continuation automaton*.

The different morphic links between these four automata have been studied too: Illie and Yu showed that the follow automaton is a quotient of the position automaton; Champarnaud and Ziadi proved that the

* D. Ziadi was supported by the MESRS - Algeria under Project 8/U03/7015.

position automaton and the c-continuation one are isomorphic and that the equation automaton is a quotient of the position one. Finally, using a join of the two previously defined quotient, Garcia *et al.* presented in [7] an automaton that is smaller than both of the follow and the equation automata.

In this paper, we extend the study of these morphic links to different computations of tree automata. We define two new tree automata constructions, *the k-position automaton* and *the follow one*, and we study their morphic links with two other already known automata constructions, the *equation automaton* of Kuske and Meinecke [11] and our *k-C-continuation automaton* [13,14]. Notice that a position automaton and a reduced one have already been defined in [12]. However, they are not isomorphic with the automata we define in this paper.

This study is motivated by the development of a library of functions for handling rational kernels [6] in the case of trees. The first problem consists in converting a regular tree expression into a tree transducer.

Section 2 recalls basic definitions and properties of regular tree languages and regular tree expressions. In Section 3, we define two new automata computations, *the k-position automaton* and *the follow one*, and recall the definition of the *equation automaton* and of the *k-C-continuation automaton*; we also present the morphic links between these four methods in this section. Section 4 is devoted to the comparison of the follow automaton and of the equation one; it is proved that there are no morphic link between them. Moreover, we extend the computation of the Garcia *et al.* equivalence leading to a smaller automaton in this section.

2 Preliminaries

Let (Σ, ar) be a *ranked alphabet*, where Σ is a finite set and ar represents the *rank* of Σ which is a mapping from Σ into \mathbb{N} . The set of symbols of rank n is denoted by Σ_n . The elements of rank 0 are called *constants*. A *tree* t over Σ is inductively defined as follows: $t = a$, $t = f(t_1, \dots, t_k)$ where a is any symbol in Σ_0 , k is any integer satisfying $k \geq 1$, f is any symbol in Σ_k and t_1, \dots, t_k are any k trees over Σ . We denote by T_Σ the set of trees over Σ . A *tree language* is a subset of T_Σ . Let $\Sigma_{\geq 1} = \Sigma \setminus \Sigma_0$ denote the set of *non-constant symbols* of the ranked alphabet Σ .

A *Finite Tree Automaton* (FTA) [5,11] \mathcal{A} is a tuple (Q, Σ, Q_T, Δ) where Q is a finite set of states, $Q_T \subset Q$ is the set of *final states* and $\Delta \subset \bigcup_{n \geq 0} (Q \times \Sigma_n \times Q^n)$ is the set of *transition rules*. This set is equivalent to the function Δ from $Q^n \times \Sigma_n \rightarrow 2^Q$ defined by $(q, f, q_1, \dots, q_n) \in \Delta \Leftrightarrow q \in \Delta(q_1, \dots, q_n, f)$. The domain of this function can be extended to $(2^Q)^n \times \Sigma_n \rightarrow 2^Q$ as follows: $\Delta(Q_1, \dots, Q_n, f) = \bigcup_{(q_1, \dots, q_n) \in Q_1 \times \dots \times Q_n} \Delta(q_1, \dots, q_n, f)$. Finally, we denote by Δ^* the function from $T_\Sigma \rightarrow 2^Q$ defined for any tree in T_Σ as follows: $\Delta^*(t) = \Delta(a)$ if $t = a$ with $a \in \Sigma_0$, $\Delta^*(t) = \Delta(f, \Delta^*(t_1), \dots, \Delta^*(t_n))$ if $t = f(t_1, \dots, t_n)$ with $f \in \Sigma_n$ and $t_1, \dots, t_n \in T_\Sigma$. A tree is *accepted* by \mathcal{A} if and only if $\Delta^*(t) \cap Q_T \neq \emptyset$. The *language recognized* by $\mathcal{L}(\mathcal{A})$ is the set of trees accepted by \mathcal{A} i.e. $\mathcal{L}(\mathcal{A}) = \{t \in T_\Sigma \mid \Delta^*(t) \cap Q_T \neq \emptyset\}$.

Let \sim be an equivalence relation over Q . We denote by $[q]$ the equivalence class of any state q in Q . The *quotient* of \mathcal{A} w.r.t. \sim is the tree automaton $\mathcal{A}/\sim = (Q/\sim, \Sigma, Q_T/\sim, \Delta/\sim)$ where: $Q/\sim = \{[q] \mid q \in Q\}$, $Q_T/\sim = \{[q] \mid q \in Q_T\}$, $\Delta/\sim = \{([q], f, [q_1], \dots, [q_n]) \mid (q, f, q_1, \dots, q_n) \in \Delta\}$.

For any integer $n \geq 0$, for any n languages $L_1, \dots, L_n \subset T_\Sigma$, and for any symbol $f \in \Sigma_n$, $f(L_1, \dots, L_n)$ is the tree language $\{f(t_1, \dots, t_n) \mid t_i \in L_i\}$. The *tree substitution* of a constant c in Σ by a language $L \subset T_\Sigma$ in a tree $t \in T_\Sigma$, denoted by $t\{c \leftarrow L\}$, is the language inductively defined by L if $t = c$; $\{d\}$ if $t = d$ where $d \in \Sigma_0 \setminus \{c\}$; $f(t_1\{c \leftarrow L\}, \dots, t_n\{c \leftarrow L\})$ if $t = f(t_1, \dots, t_n)$ with $f \in \Sigma_n$ and t_1, \dots, t_n any n trees over Σ . Let c be a symbol in Σ_0 . The *c-product* $L_1 \cdot c L_2$ of two languages $L_1, L_2 \subset T_\Sigma$ is defined by $L_1 \cdot c L_2 = \bigcup_{t \in L_1} \{t\{c \leftarrow L_2\}\}$. The *iterated c-product* is inductively defined for $L \subset T_\Sigma$ by: $L^{0c} = \{c\}$ and $L^{(n+1)c} = L^{nc} \cup L \cdot c L^{nc}$. The *c-closure* of L is defined by $L^{*c} = \bigcup_{n \geq 0} L^{nc}$.

A *regular expression* over a ranked alphabet Σ is inductively defined by $E = 0$, $E \in \Sigma_0$, $E = f(E_1, \dots, E_n)$, $E = (E_1 + E_2)$, $E = (E_1 \cdot c E_2)$, $E = (E_1^{*c})$, where $c \in \Sigma_0$, $n \in \mathbb{N}$, $f \in \Sigma_n$ and E_1, E_2, \dots, E_n are any n regular expressions over Σ . Parenthesis can be omitted when there is no ambiguity. We write $E_1 = E_2$ if E_1 and E_2 graphically coincide. We denote by $\text{RegExp}(\Sigma)$ the set of all regular expressions over Σ . Every regular expression E can be seen as a tree over the ranked alphabet $\Sigma \cup \{+, \cdot_c, *c\}$ with $c \in \Sigma_0$ where $+$ and

\cdot_c can be seen as a symbol of rank 2 and \ast_c has rank 1. This tree is the syntax-tree T_E of E . The *alphabetical width* $\|E\|$ of E is the number of occurrences of symbols of Σ in E . The *size* $|E|$ of E is the size of its syntax tree T_E .

The language $\llbracket E \rrbracket$ denoted by E is inductively defined as $\llbracket 0 \rrbracket = \emptyset$, $\llbracket c \rrbracket = \{c\}$, $\llbracket f(E_1, E_2, \dots, E_n) \rrbracket = f(\llbracket E_1 \rrbracket, \dots, \llbracket E_n \rrbracket)$, $\llbracket E_1 + E_2 \rrbracket = \llbracket E_1 \rrbracket \cup \llbracket E_2 \rrbracket$, $\llbracket E_1 \cdot_c E_2 \rrbracket = \llbracket E_1 \rrbracket \cdot_c \llbracket E_2 \rrbracket$, $\llbracket E_1^{\ast_c} \rrbracket = \llbracket E_1 \rrbracket^{\ast_c}$ where $n \in \mathbb{N}$, E_1, E_2, \dots, E_n are any n regular expressions, $f \in \Sigma_n$ and $c \in \Sigma_0$. It is well known that a tree language is accepted by some tree automaton if and only if it can be denoted by a regular expression [5, 11].

A regular expression E defined over Σ is *linear* if and only if every symbol of $\Sigma_{\geq 1}$ appears at most once in E . Note that any constant symbol may occur more than once. Let E be a regular expression over Σ . The *linearized regular expression* \bar{E} in E of a regular expression E is obtained from E by marking differently all symbols of a rank greater than or equal to 1 (symbols of $\Sigma_{\geq 1}$). The set of *marked symbols* with symbols of Σ_0 is the ranked alphabet containing symbols called *positions*. We denote this set by $\text{Pos}_E(E)$.

The mapping h is defined from $\text{Pos}_E(E)$ to Σ with $h(\text{Pos}_E(E)_m) \subset \Sigma_m$ for every $m \in \mathbb{N}$. It associates with a marked symbol $f_j \in \text{Pos}_E(E)_{\geq 1}$ the symbol $f \in \Sigma_{\geq 1}$ and for a symbol $c \in \Sigma_0$ the symbol $h(c) = c$. We can extend the mapping h naturally to $\text{RegExp}(\text{Pos}_E(E)) \rightarrow \text{RegExp}(\Sigma)$ by $h(a) = a$, $h(E_1 + E_2) = h(E_1) + h(E_2)$, $h(E_1 \cdot_c E_2) = h(E_1) \cdot_c h(E_2)$, $h(E_1^{\ast_c}) = h(E_1)^{\ast_c}$, $h(f_j(E_1, \dots, E_n)) = f(h(E_1), \dots, h(E_n))$, with $n \in \mathbb{N}$, $a \in \Sigma_0$, $f \in \Sigma_n$, $f_j \in \text{Pos}_E(E)_n$ such that $h(f_j) = f$ and E_1, \dots, E_n any regular expressions over $\text{Pos}_E(E)$.

3 Tree Automata from Regular Expressions

In this section, we show how to compute from a regular expression E four tree automata accepting $\llbracket E \rrbracket$: we introduce two new constructions, the K -position automaton and the follow automaton of E , and then we recall two already-known constructions, the equation automaton [11] and the C-continuation one [13].

Regular languages defined over ranked alphabet Σ are exactly the languages denoted by a regular expression on Σ . There may exist many distinct regular expressions which denote the same regular language. Two regular expressions are said to be *equivalent* if they denote the same language. To simplify handling about regular expressions, we define *trivial identities* for which regular expressions are equivalent in an obvious way. Let $E_1 \dots E_n$ be n regular expressions over a ranked alphabet Σ and c be a symbol in Σ_0 . It can be trivially shown that:

$$\llbracket E_1 + 0 \rrbracket = \llbracket 0 + E_1 \rrbracket = \llbracket E_1 \rrbracket, \llbracket E_1 \cdot_c 0 \rrbracket = \llbracket 0 \cdot_c E_1 \rrbracket = \llbracket 0 \rrbracket, \llbracket 0^{\ast_c} \rrbracket = \llbracket c \rrbracket, \llbracket f(E_1, \dots, 0, \dots, E_n) \rrbracket = \llbracket 0 \rrbracket.$$

Consequently, we consider that:

$$E_1 + 0 = 0 + E_1 = E_1, E_1 \cdot_c 0 = 0 \cdot_c E_1 = 0, 0^{\ast_c} = c, f(E_1, \dots, 0, \dots, E_n) = 0.$$

It is easy to see that in these reduced expressions, either there is no occurrences of 0 in E or $E = 0$.

In the following of this section, E is any regular expression over a ranked alphabet Σ .

3.1 The K -Position Tree Automaton

In this section, we show how to compute the K -position tree automaton of a regular expression E , recognizing $\llbracket E \rrbracket$. This is an extension of the well-known position automaton [8] for word regular expressions. Its computation is based on the computations of particular *position functions*, defined in the following.

In what follows, for any two trees s and t , we denote by $s \preceq t$ the relation " s is a subtree of t ". Let k be an integer and $t = f(t_1, \dots, t_n)$ be a tree. We denote by $\text{root}(t)$ the root of t , by $k\text{-child}(t)$ the k^{th} child of f in t , that is the root of t_k if it exists, and by $\text{Leaves}(t)$ the set of the leaves of t , i.e. $\{s \in \Sigma_0 \mid s \preceq t\}$.

Let E be linear, $1 \leq k \leq m$ be two integers and f be a symbol in Σ_m . The set $\text{First}(E)$ is the subset of Σ defined by $\{\text{root}(t) \in \Sigma \mid t \in \llbracket E \rrbracket\}$; The set $\text{Follow}(E, f, k)$ is the subset of Σ defined by $\{g \in \Sigma \mid \exists t \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = g\}$; The set $\text{Last}(E)$ is the subset of Σ_0 defined by $\text{Last}(E) = \bigcup_{t \in \llbracket E \rrbracket} \text{Leaves}(t)$.

Example 1. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ be defined by $\Sigma_0 = \{a, b, c\}$, $\Sigma_1 = \{f, h\}$ and $\Sigma_2 = \{g\}$. Let us consider the regular expression E and its linearized form defined by:

$$\begin{aligned} E &= (f(a)^{*a} \cdot_a b + h(b))^{*b} + g(c, a)^{*c} \cdot_c (f(a)^{*a} \cdot_a b + h(b))^{*b}, \\ \bar{E} &= (f_1(a)^{*a} \cdot_a b + h_2(b))^{*b} + g_3(c, a)^{*c} \cdot_c (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b}. \end{aligned}$$

The language denoted by \bar{E} is $\llbracket \bar{E} \rrbracket = \{b, f_1(b), f_1(f_1(b)), f_1(h_2(b)), h_2(b), h_2(f_1(b)), h_2(h_2(b)), \dots, g_3(b, a), g_3(g_3(b), a), g_3(f_4(b), a), g_3(h_5(b), a), f_4(f_4(b)), f_4(h_5(b), h_5(f_4(b)), h_5(h_5(b)), \dots\}$.

Consequently, $\text{First}(\bar{E}) = \{b, f_1, h_2, g_3, f_4, h_5\}$ and $\text{Follow}(\bar{E}, f_1, 1) = \{b, f_1, h_2\}$, $\text{Follow}(\bar{E}, h_2, 1) = \{b, f_1, h_2\}$, $\text{Follow}(\bar{E}, g_3, 1) = \{b, g_3, f_4, h_5\}$, $\text{Follow}(\bar{E}, g_3, 2) = \{a\}$, $\text{Follow}(\bar{E}, f_4, 1) = \{b, f_4, h_5\}$, $\text{Follow}(\bar{E}, h_5, 1) = \{b, f_4, h_5\}$.

Let us first show that the position functions First and Follow are directly computable from a regular expression over its syntactic tree.

Lemma 1. *Let E be linear. The set $\text{First}(E)$ can be computed inductively as follows:*

$$\begin{aligned} \text{First}(0) &= \emptyset, \\ \text{First}(a) &= \{a\}, \\ \text{First}(f(E_1, \dots, E_m)) &= \{f\}, \\ \text{First}(E_1 + E_2) &= \text{First}(E_1) \cup \text{First}(E_2), \text{First}(E_1^{*c}) = \text{First}(E_1) \cup \{c\}, \\ \text{First}(E_1 \cdot_c E_2) &= \begin{cases} (\text{First}(E_1) \setminus \{c\}) \cup \text{First}(E_2) & \text{if } c \in \llbracket E_1 \rrbracket, \\ \text{First}(E_1) & \text{otherwise.} \end{cases} \end{aligned}$$

Proof. Let us show by induction over E that any symbol f in Σ belongs to $\text{First}(E)$ if and only if there exists a tree t in $\llbracket E \rrbracket$ the root of which is f (i.e. $t = f(t_1, \dots, t_n)$).

1. If $E = 0$, $\llbracket E \rrbracket = \emptyset = \text{First}(E)$.
2. Let us suppose that $E = a$ with $a \in \Sigma_0$. Hence $\llbracket E \rrbracket = \{a\} = \text{First}(E)$.
3. Suppose that $E = f(E_1, \dots, E_n)$. Hence $\llbracket E \rrbracket = f(\llbracket E_1 \rrbracket, \dots, \llbracket E_n \rrbracket)$. As a direct consequence, any tree t in $\llbracket E \rrbracket$ admits f as root. Furthermore, since we consider trivial identities, $E \neq 0 \Leftrightarrow \llbracket E \rrbracket \neq \emptyset$. Consequently, $\{f\} = \text{First}(E)$ and $\exists f(t_1, \dots, t_n) \in \llbracket E \rrbracket$.
4. Suppose that $E = E_1 + E_2$. Then:

$$\begin{aligned} f \in \text{First}(E_1) \cup \text{First}(E_2) &\Leftrightarrow f \in \text{First}(E_1) \vee f \in \text{First}(E_2) \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \vee \exists f(t_1, \dots, t_n) \in \llbracket E_2 \rrbracket \quad \textbf{(Induction Hypothesis)} \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \cup \llbracket E_2 \rrbracket \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E \rrbracket \end{aligned}$$
5. Let us consider that $E = E_1 \cdot_c E_2$, with $c \in \Sigma_0$.

(a) If $c \notin \llbracket E_1 \rrbracket$, then $\text{First}(E) = \text{First}(E_1)$.

Let $t = f(t_1, \dots, t_n)$ be a tree in $\llbracket E_1 \rrbracket \setminus \{c\}$. Since $\llbracket E_2 \rrbracket \neq \emptyset$, $t_{c \leftarrow \llbracket E_2 \rrbracket} \neq \emptyset$. Furthermore, any tree t' in $t_{c \leftarrow \llbracket E_2 \rrbracket} \neq \emptyset$ admits the same root as t by construction. Hence $\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \Rightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \cdot_c E_2 \rrbracket$.

Reciprocally, let $t' = f(t'_1, \dots, t'_n)$ be a tree in $\llbracket E_1 \cdot_c E_2 \rrbracket$. By definition, there exists a tree t in $\llbracket E_1 \rrbracket$ such that $t' \in t_{c \leftarrow \llbracket E_2 \rrbracket}$. Trivially, since $t \neq c$, t and t' admit the same root. Hence $\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \cdot_c E_2 \rrbracket$.

Consequently, the following condition **(C1)** holds:

$$\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \cdot_c \llbracket E_2 \rrbracket.$$

Finally:

$$\begin{aligned} f \in \text{First}(E) &\Leftrightarrow f \in \text{First}(E_1) \\ &\Leftrightarrow \exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \quad \textbf{(Induction hypothesis)} \\ &\Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \cdot_c \llbracket E_2 \rrbracket \quad \textbf{(C1)} \\ &\Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E \rrbracket \end{aligned}$$

- (b) Consider that $c \in \llbracket E_1 \rrbracket$. Hence $\text{First}(E) = \text{First}(E_1) \setminus \{c\} \cup \text{First}(E_2)$. Since $E \neq 0 \Rightarrow E_2 \neq 0 \Rightarrow \llbracket E_2 \rrbracket \neq \emptyset$, let u_2 be a tree in $\llbracket E_2 \rrbracket$. Similarly, since $E \neq 0 \Rightarrow E_1 \neq 0 \Rightarrow \llbracket E_1 \rrbracket \neq \emptyset$, let u_1 be a tree in $\llbracket E_1 \rrbracket$.

According to condition **(C1)** in case 5a,

$$\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \cdot_c \llbracket E_2 \rrbracket.$$

Furthermore, by definition of \cdot_c , the following condition **(C2)** holds:

$$\exists f(t_1, \dots, t_n) \in \llbracket E_2 \rrbracket \Leftrightarrow \exists f(t_1, \dots, t_n) \in \{c\} \cdot_c \llbracket E_2 \rrbracket$$

Consequently:

$$\begin{aligned} f \in \text{First}(E) &\Leftrightarrow f \in \text{First}(E_1) \setminus \{c\} \cup \text{First}(E_2) \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \vee \exists f(t_1, \dots, t_n) \in \llbracket E_2 \rrbracket \quad \textbf{(Induction hypothesis)} \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \cdot_c \llbracket E_2 \rrbracket \vee \exists f(t_1, \dots, t_n) \in \{c\} \cdot_c \llbracket E_2 \rrbracket \quad \textbf{(C1) and (C2)} \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \cdot_c \llbracket E_2 \rrbracket \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E \rrbracket \end{aligned}$$

6. Let us suppose that $E = E_1^{*c}$. Then $\text{First}(E) = \text{First}(E_1) \cup \{c\}$. Let f be a symbol in Σ . If $f = c$, then $c \in \text{First}(E)$ and $c \in \llbracket E \rrbracket$. Otherwise, since $E \neq 0 \Rightarrow E_1 \neq 0 \Rightarrow \llbracket E_1 \rrbracket \neq \emptyset$, there exists some tree in $\llbracket E_1 \rrbracket$.

According to condition **(C1)** in case 5a,

$$\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \cdot_c \llbracket E_1 \rrbracket.$$

By successive application of **(C1)**, it can be shown that the following condition **(C3)** holds: for any integer $n \geq 1$,

$$\exists t = f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \Leftrightarrow \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket^{n_c}.$$

Consequently:

$$\begin{aligned} f \in \text{First}(E) &\Leftrightarrow f \in \text{First}(E_1) \setminus \{c\} \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1 \rrbracket \setminus \{c\} \quad \textbf{(Induction hypothesis)} \\ &\Leftrightarrow \exists n \geq 1, \exists t' = f(t'_1, \dots, t'_n) \in \llbracket E_1 \rrbracket^{n_c} \\ &\Leftrightarrow \exists f(t_1, \dots, t_n) \in \llbracket E_1^{*c} \rrbracket \end{aligned}$$

Lemma 2. Let E be linear, $1 \leq k \leq m$ be two integers and f be a symbol in Σ_m . The set of symbols $\text{Follow}(E, f, k)$ can be computed inductively as follows:

$$\begin{aligned} \text{Follow}(0, f, k) &= \text{Follow}(a, f, k) = \emptyset, \\ \text{Follow}(g(E_1, \dots, E_n), f, k) &= \begin{cases} \text{First}(E_k) & \text{if } f = g, \\ \text{Follow}(E_l, f, k) & \text{if } \exists l \mid f \in \Sigma_{E_l}, \\ \emptyset & \text{otherwise.} \end{cases} \\ \text{Follow}(E_1 + E_2, f, k) &= \begin{cases} \text{Follow}(E_1, f, k) & \text{if } f \in \Sigma_{E_1}, \\ \text{Follow}(E_2, f, k) & \text{if } f \in \Sigma_{E_2}, \\ \emptyset & \text{otherwise.} \end{cases} \\ \text{Follow}(E_1 \cdot_c E_2, f, k) &= \begin{cases} (\text{Follow}(E_1, f, k) \setminus \{c\}) \cup \text{First}(E_2) & \text{if } c \in \text{Follow}(E_1, f, k), \\ \text{Follow}(E_1, f, k) & \text{if } f \in \Sigma_{E_1} \wedge c \notin \text{Follow}(E_1, f, k), \\ \text{Follow}(E_2, f, k) & \text{if } f \in \Sigma_{E_2} \wedge c \in \text{Last}(E_1), \\ \emptyset & \text{otherwise,} \end{cases} \\ \text{Follow}(E_1^{*c}, f, k) &= \begin{cases} \text{Follow}(E_1, f, k) \cup \text{First}(E_1) & \text{if } c \in \text{Follow}(E_1, f, k), \\ \text{Follow}(E_1, f, k) & \text{otherwise,} \end{cases} \end{aligned}$$

Proof. Let F be any linear regular expression, h be any symbol in Σ , and j be any integer. We denote by $\phi_F^{h,j}$ the set $\{g \in \Sigma \mid \exists t \in \llbracket F \rrbracket, \exists s \preceq t, \text{root}(s) = h \wedge j\text{-child}(s) = g\}$. Let us show by induction over the structure of E that $\text{Follow}(E, f, k) = \phi_E^{f,k}$. Let h be a symbol in Σ .

1. If $E = 0$ or if $E = a$, then $\text{Follow}(E, f, k) = \emptyset = \phi_E^{f,k}$.
2. Suppose that $E = g(E_1, \dots, E_n)$. Since $E \neq 0$, then for any $1 \leq j \leq n$, $E_j \neq 0$ and then there exists at least one tree u_j in $\llbracket E_j \rrbracket$. Three cases have to be considered:
 - (a) if $f = g$, since E is linear, for any integer $j \in \{1, \dots, n\}$ it holds that $f \notin \Sigma_{E_j}$. Then:

$$\begin{aligned} h \in \text{Follow}(E, f, k) &\Leftrightarrow h \in \text{First}(E_k) \\ &\Leftrightarrow \exists t = h(t_1, \dots, t_n) \in \llbracket E_k \rrbracket \quad \textbf{(Lemma 1)} \\ &\Leftrightarrow \exists t' = f(u_1, \dots, u_{k-1}, h(t_1, \dots, t_n), \dots, u_n) \in \llbracket E \rrbracket \\ &\Leftrightarrow h \in \phi_E^{f,k} \end{aligned}$$

(b) Consider that there exists an integer l such that $f \in \Sigma_{E_l}$. Then:

$$\begin{aligned}
h \in \text{Follow}(E, f, k) &\Leftrightarrow h \in \text{Follow}(E_l, f, k) \\
&\Leftrightarrow h \in \phi_{E_l}^{f,k} \quad \textbf{(Induction hypothesis)} \\
&\Leftrightarrow \exists t \in \llbracket E_l \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\Leftrightarrow \exists t' = g(u_1, \dots, u_{l-1}, t, \dots, u_n) \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\Leftrightarrow \exists t' \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\Leftrightarrow h \in \phi_E^{f,k}
\end{aligned}$$

(c) Otherwise, since $f \notin \Sigma_E$, $\text{Follow}(E, f, k) = \emptyset = \phi_E^{f,k}$.

3. Suppose that $E = E_1 + E_2$. Then:

$$\begin{aligned}
h \in \text{Follow}(E, f, k) &\Leftrightarrow h \in \text{Follow}(E_1, f, k) \vee h \in \text{Follow}(E_2, f, k) \\
&\Leftrightarrow h \in \phi_{E_1}^{f,k} \vee h \in \phi_{E_2}^{f,k} \quad \textbf{(Induction hypothesis)} \\
&\Leftrightarrow \exists t \in \llbracket E_1 \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\quad \vee \exists t \in \llbracket E_2 \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\Leftrightarrow \exists t \in \llbracket E_1 \rrbracket \cup \llbracket E_2 \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h \\
&\Leftrightarrow h \in \phi_E^{f,k}
\end{aligned}$$

4. Let us consider that $E = E_1 \cdot_c E_2$. The following cases have to be considered.

(a) Let us suppose that c is in $\text{Follow}(E_1, f, k)$.

- i. Suppose that h is in $\text{Follow}(E_1, f, k) \setminus \{c\}$. According to induction hypothesis, h belongs to $\phi_{E_1}^{f,k}$ and then there exists a tree t in $\llbracket E_1 \rrbracket$ that contains a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = h$. Since $h \neq c$, any tree t' in $t_{c \leftarrow \llbracket E_2 \rrbracket}$ still contains the subtree s . Consequently, there exists a tree t' in $\llbracket E \rrbracket$ that contains a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = h$, i.e. h is in $\phi_E^{f,k}$.
- ii. Suppose now that h is in $\text{First}(E_2)$. According to Lemma 1, there exists a tree $t' = h(t_1, \dots, t_n)$ in $\llbracket E_2 \rrbracket$. Since c is in $\text{Follow}(E_1, f, k)$, then by induction hypothesis c belongs to $\phi_{E_1}^{f,k}$; therefore there exists a tree t_c in $\llbracket E_1 \rrbracket$ containing a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = c$. Consequently, $(t_c)_{c \leftarrow t'}$ admits by construction a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = c$. Therefore h is in $\phi_E^{f,k}$.
- iii. Conversely, suppose that $h \in \phi_E^{f,k}$. Then there exists a tree t' in $\llbracket E \rrbracket$ that contains a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = h$. By definition of $\llbracket E \rrbracket$, t' is in $t_{c \leftarrow \llbracket E_2 \rrbracket}$ with t a tree in $\llbracket E_1 \rrbracket$. If h is in $\Sigma_{E_1} \setminus \{c\}$, since E is linear, h appears in t and so do f and s . Consequently, h is in $\text{Follow}(E_1, f, k) \subset \text{Follow}(E, f, k)$. If h is in Σ_{E_2} , then the subtree s is created by substituting to a leave c whose father is f a tree in $\llbracket E_2 \rrbracket$ the root of which is h . Consequently h is in $\text{First}(E_2) \subset \text{Follow}(E, f, k)$.

(b) Suppose now that $f \in \Sigma_{E_1}$ and that $c \notin \text{Follow}(E_1, f, k)$.

- i. Consider that $h \in \text{Follow}(E_1, f, k) \setminus \{c\}$. According to case 4(a)i, $h \in \phi_E^{f,k}$.
- ii. Conversely, suppose that $h \in \phi_E^{f,k}$. Then $\exists t' \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h$. By definition, $t' \in t_{c \leftarrow \llbracket E_2 \rrbracket}$ with $t \in \llbracket E_1 \rrbracket$. Since $c \notin \text{Follow}(E_1, f, k)$, $h \in \Sigma_{E_1} \setminus \{c\}$ and since E is linear, h appears in t and so do f and s . Consequently, $h \in \text{Follow}(E_1, f, k) = \text{Follow}(E, f, k)$.

(c) Consider that $f \in \Sigma_{E_2}$ and that $c \in \text{Last}(E_1)$. By definition, there exists a tree $t_c \in \llbracket E_1 \rrbracket$ such that $c \in \text{Leaves}(t_c)$.

- i. Suppose that $h \in \text{Follow}(E_2, f, k)$. By induction hypothesis, $\exists t \in \llbracket E_2 \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. Moreover, s is a subtree of $(t_c)_{c \leftarrow t} \subset (t_c)_{c \leftarrow \llbracket E_2 \rrbracket} \subset \llbracket E \rrbracket$. Consequently, $\exists t \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. Hence, $h \in \phi_E^{f,k}$.
- ii. Conversely, suppose that $h \in \phi_E^{f,k}$. Then $\exists t' \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h$. By definition, $t' \in t_{c \leftarrow \llbracket E_2 \rrbracket}$ with $t \in \llbracket E_1 \rrbracket$. Since $f \in \Sigma_{E_2}$, so does h . Consequently, s is a subset that belongs to a tree t'' that has been substituted to a leave c of t to appear in t' . Hence $h \in \text{Follow}(E_2, f, k) = \text{Follow}(E, f, k)$.

(d) In all other cases, there is no tree t in $\llbracket E \rrbracket$ such that $\exists s \preceq t, \text{root}(s) = f \wedge k\text{-child}(s) = h$. Hence $\text{Follow}(E, f, k) = \emptyset = \phi_E^{f,k}$.

5. Let us consider that $E = E_1^{*c}$.
 - (a) Consider that $h \in \text{Follow}(E_1, f, k)$. By induction hypothesis, $\exists t \in \llbracket E_1 \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. Since $\llbracket E_1 \rrbracket \subset \llbracket E \rrbracket$, $\exists t \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. Hence $h \in \phi_E^{f,k}$.
 - (b) Suppose that $h \in \text{First}(E_1)$. According to Lemma 1, there exists a tree $t' = h(t_1, \dots, t_n)$ in $\llbracket E_1 \rrbracket$. By construction, $c \in \text{Follow}(E_1, f, k)$. By induction hypothesis, c belongs to $\phi_{E_1}^{f,k}$ and then there exists a tree t_c in $\llbracket E_1 \rrbracket$ that contains a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = c$. Moreover, $t = (t_c)_{c \leftarrow t'}$ belongs by construction to $\llbracket E_1 \rrbracket \cdot_c \llbracket E_1 \rrbracket \subset \llbracket E_1 \rrbracket^{*c}$ and contains a subtree s such that $\text{root}(s) = f$ and $k\text{-child}(s) = h$. Therefore $\exists t \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. Hence $h \in \phi_E^{f,k}$.
 - (c) Conversely, suppose that $h \in \phi_E^{f,k}$. Then $\exists t \in \llbracket E \rrbracket, \exists s \preceq t, \text{root}(s) = f, k\text{-child}(s) = h$. By construction, $t \in t'_{c \leftarrow \llbracket E_1 \rrbracket}$ with t' a tree in $\llbracket E_1 \rrbracket^{n_c}$ for some integer n . Let us consider that t is the tree satisfying the previous condition with minimal n . If s appears in t' , contradiction with the minimality of n . By construction, t is obtained from t' by substituting to its leaves c a tree u in $\llbracket E_1 \rrbracket$. If s is a subtree of u , $h \in \phi_{E_1}^{f,k}$ and by induction hypothesis, $h \in \text{Follow}(E_1, f, k) \subset \text{Follow}(E, f, k)$. If s does not appear in u , then it is created during the c -product, i.e. t' admits a leave c the root of which is f , and u admits h as root. Consequently, by induction hypothesis, $c \in \text{Follow}(E_1, f, k)$ and according to Lemma 1, $h \in \text{First}(E_1) \subset \text{Follow}(E, f, k)$.

The two functions First and Follow are sufficient to compute the K -position tree automaton of E .

Definition 1. Let E be linear. The K -position automaton \mathcal{P}_E is the automaton (Q, Σ, Q_T, Δ) defined by

$$Q = \{f^k \mid f \in \Sigma_m \wedge 1 \leq k \leq m\} \cup \{\varepsilon^1 \mid \varepsilon^1 \text{ is a new symbol not in } \Sigma\}, \quad Q_T = \{\varepsilon^1\},$$

$$\Delta = \begin{cases} \{(f^k, g, g^1, \dots, g^n) \mid g \in \Sigma_n \wedge g \in \text{Follow}(E, f, k)\} \\ \cup \{(\varepsilon^1, f, f^1, \dots, f^m) \mid f \in \Sigma_m \wedge f \in \text{First}(E)\} \\ \cup \{(\varepsilon^1, c) \mid c \in \Sigma_0 \wedge c \in \text{First}(E)\} \\ \cup \{(f^k, c) \mid c \in \text{Follow}(E, f, k)\} \end{cases}$$

In order to show that the K -position tree automaton of E accepts $\llbracket E \rrbracket$, we characterize the membership of a tree t in the language denoted by E using the functions First and Follow.

Proposition 1. Let E be linear and t be a term in T_Σ . The tree t belongs to $\llbracket E \rrbracket$ if and only if the two following conditions are satisfied:

1. the root of t is in $\text{First}(E)$,
2. for any subtree $f(t_1, \dots, t_m)$ of t , for any integer k in $\{1, \dots, m\}$, the root of t_k belongs to $\text{Follow}(E, f, k)$.

Proof. If t belongs to $\llbracket E \rrbracket$, it holds by definition that $\text{root}(t) \in \text{First}(E)$ and that $\forall f(t_1, \dots, t_m) \preceq t, \forall 1 \leq k \leq m, \text{root}(t_k) \in \text{Follow}(E, f, k)$. Let us show the reciprocal part by induction over the structure of E , i.e. **(I)** $\text{root}(t) \in \text{First}(E)$ and **(II)** $\forall f(t_1, \dots, t_m) \preceq t, \forall 1 \leq k \leq m, \text{root}(t_k) \in \text{Follow}(E, f, k) \Rightarrow t \in \llbracket E \rrbracket$.

1. If $E = a$, $\text{First}(E) = \{a\}$. Hence $\text{root}(t) \in \text{First}(E) \Rightarrow t = a \Rightarrow t \in \llbracket E \rrbracket$.
2. Let us consider that $E = g(E_1, \dots, E_n)$. In this case, $\text{First}(E) = \{g\}$. From **(I)**, $\text{root}(t) = g \in \text{First}(E)$, and then $t = g(u_1, \dots, u_n)$. Let $u_j \in \{u_1, \dots, u_n\}$. From **(II)** $\text{root}(u_j) \in \text{Follow}(E, f, j) = \text{First}(E_j)$ **(A)**. Since u_j is a subtree of t , Condition **(II)** also implies that $\forall f(t_1, \dots, t_m) \preceq u_j, \forall 1 \leq k \leq m, \text{root}(t_k) \in \text{Follow}(E, f, k)$ **(B)**. Since f appears in u_j , $f \in \Sigma_{E_j}$ and then $\text{Follow}(E, f, k) = \text{Follow}(E_j, f, k)$ **(C)**. By induction hypothesis, the conditions **(A)**, **(B)** and **(C)** implies that $u_j \in \llbracket E_j \rrbracket$. Consequently $t = g(u_1, \dots, u_n) \in \llbracket E \rrbracket$.
3. Let us suppose that $E = E_1 + E_2$. Then:

$$\begin{aligned} & \text{root}(t) \in \text{First}(E), \forall f(t_1, \dots, t_m) \preceq t, \forall 1 \leq k \leq m, \text{root}(t_k) \in \text{Follow}(E, f, k) \\ & \Rightarrow \text{root}(t) \in \text{First}(E_j), \forall f(t_1, \dots, t_m) \preceq t, \forall 1 \leq k \leq m, \text{root}(t_k) \in \text{Follow}(E_j, f, k), j \in \{1, 2\} \\ & \Rightarrow t \in \llbracket E_j \rrbracket, j \in \{1, 2\} \quad \textbf{(Induction hypothesis)} \\ & \Rightarrow t \in \llbracket E_1 \rrbracket \cup \llbracket E_2 \rrbracket \end{aligned}$$

4. Let us consider that $E = E_1 \cdot_c E_2$. Two cases have to be considered.

- (a) If $\text{root}(t)$ is in $\text{First}(E_2)$, then c is in $\llbracket E_1 \rrbracket$. Consequently, any symbol appearing in t belongs to Σ_{E_2} by definition of Follow and following **(II)**. Hence, for any subtree $f(t_1, \dots, t_m)$ of t , for any integer k in $\{1, \dots, m\}$, the root of t_k is in $\text{Follow}(E_2, f, k)$. By induction hypothesis, $t \in \llbracket E_2 \rrbracket$. Furthermore $\llbracket E_2 \rrbracket = c \cdot_c \llbracket E_2 \rrbracket \subset \llbracket E_1 \rrbracket \cdot_c \llbracket E_2 \rrbracket$. Consequently t is in $\llbracket E_1 \rrbracket \cdot_c \llbracket E_2 \rrbracket$.
- (b) Suppose that $\text{root}(t)$ is in $\text{First}(E_1)$. Necessarily, by definition of First, the root of t cannot be c .

Let us construct the tree $u = \phi(t)$ where ϕ is inductively defined for any tree t' as follows:

$$\phi(t') = \begin{cases} f(\phi(t_1), \dots, \phi(t_n)) & \text{if } t' = f(t_1, \dots, t_n) \wedge (f = \text{root}(t) \vee f \in \Sigma_{E_1}), \\ c & \text{otherwise.} \end{cases}$$

Notice that a subtree s of t is substituted by c when its root is in $\text{First}(E_2)$. Therefore, if s is the k -th child of a node f , then it holds $c \in \text{Follow}(E_1, f, k)$.

Since by construction, **(I)** implies $\text{root}(u) = \text{root}(t)$ is in $\text{First}(E_1)$ and **(II)** implies $\forall f(t_1, \dots, t_m) \preceq u, \forall 1 \leq k \leq m$, $\text{root}(t_k)$ is in $\text{Follow}(E_1, f, k)$, it holds according to induction hypothesis that $u \in \llbracket E_1 \rrbracket$. Furthermore, let us consider the set $V = \{v \mid g(r_1, \dots, v, \dots, r_l) \preceq t \wedge \text{root}(v) \in \Sigma_{E_2} \wedge g \in \Sigma_{E_1}\}$. Let v be a tree in V . By construction, v satisfies $\text{root}(v) \in \text{First}(E_2)$ since it is in $\text{Follow}(E, g, x)$ for some integer x . Moreover, **(II)** implies that $\forall f(t_1, \dots, t_m) \preceq v, \forall 1 \leq k \leq m$, $\text{root}(t_k)$ is in $\text{Follow}(E, f, k) = \text{Follow}(E_2, f, k)$. It holds according to induction hypothesis that $v \in \llbracket E_2 \rrbracket$. Finally, t can be obtained from u by substituting to the leaves labelled by c a tree in V . Therefore t is in $t'_{c \leftarrow V} \subset t'_{c \leftarrow \llbracket E_2 \rrbracket} \subset \llbracket E_1 \rrbracket \cdot_c \llbracket E_2 \rrbracket$.

- 5. Let us suppose that $E = E_1^{*c}$. If $\text{root}(t) = c$ then $t = c$ and $t \in \llbracket E \rrbracket$. Otherwise, $\text{root}(t)$ is in $\text{First}(E_1)$. Furthermore, **(II)** implies that $\forall f(t_1, \dots, t_m) \preceq t, \forall 1 \leq k \leq m$, $\text{root}(t_k)$ is either in $\text{First}(E_1)$ if $c \in \text{Follow}(E_1, f, k)$, or in $\text{Follow}(E_1, f, k)$.

Let us construct the tree $u = \phi(t)$ where ϕ is inductively defined for any tree t' as follows:

$$\phi(t') = \begin{cases} f(\phi(t_1), \dots, \phi(t_n)) & \text{if } t' = f(t_1, \dots, t_n) \wedge (f = \text{root}(t) \vee f \notin \text{First}(E_1)), \\ c & \text{otherwise.} \end{cases}$$

Notice that a subtree s is substituted by c when its root is in $\text{First}(E_1)$. Therefore, if s is the k -th child of a node f , then it holds $c \in \text{Follow}(E_1, f, k)$.

Since by construction, **(I)** implies that $\text{root}(u) = \text{root}(t)$ is in $\text{First}(E_1)$, and since **(II)** implies that $\forall f(t_1, \dots, t_m) \preceq u, \forall 1 \leq k \leq m$, $\text{root}(t_k)$ is in $\text{Follow}(E_1, f, k)$, it holds according to induction hypothesis that $u \in \llbracket E_1 \rrbracket$. Furthermore, let us consider the set $V = \{v \mid g(r_1, \dots, v, \dots, r_l) \preceq t \wedge \text{root}(v) \in \text{First}(E_1)\} \setminus \{t\}$. Let v be a tree in V . By construction, v satisfies $\text{root}(v) \in \text{First}(E_1)$, and $\forall f(t_1, \dots, t_m) \preceq v, \forall 1 \leq k \leq m$, either $\text{root}(t_k)$ is in $\text{First}(E_1)$ if $c \in \text{Follow}(E_1, f, k)$, or in $\text{Follow}(E_1, f, k)$.

The function ϕ can be applied over any tree v in V to produce another tree in $\llbracket E_1 \rrbracket$ and another set of smaller trees V' . By recurrence, it can be shown that this process halts and that any tree in V belongs to $\llbracket E_1 \rrbracket^{n_c}$ for some integer n . As a direct consequence, since t can be obtained from u by substituting to the leaves labelled by c a tree in V , the tree t is in $u_{c \leftarrow V} \subset u_{c \leftarrow \llbracket E_1 \rrbracket^{n_c}} \subset \llbracket E_1 \rrbracket^{(n+1)c} \subset \llbracket E_1 \rrbracket^{*c}$.

Let us show how to link the characterization in Proposition 1 with the transition sequences in \mathcal{P}_E .

Proposition 2. *Let E be linear and $\mathcal{P}_E = (Q, \Sigma, Q_T, \Delta)$. Let $t = f(t_1, \dots, t_m)$ be a term in T_Σ . Then the two following propositions are equivalent:*

1. $\forall g(s_1, \dots, s_l) \preceq t, \forall p \leq l, \text{root}(s_p) \in \text{Follow}(E, g, p)$,
2. $\forall 1 \leq k \leq m, f^k \in \Delta^*(t_k)$.

Proof. By induction over t .

- Let $t = f(a_1, \dots, a_m)$ with $a_j \in \Sigma_0$ for any integer $1 \leq j \leq m$. Hence, by construction of \mathcal{P}_E , for any integer $j \leq m$, $\text{root}(a_j) \in \text{Follow}(E, f, j) \Leftrightarrow f^j \in \Delta(a_j)$.
- Suppose that $t = f(t_1, \dots, t_m)$, with $t_j = g_j(r_{j,1}, \dots, r_{j,n_j})$ for any integer $1 \leq j \leq m$. Let j be an integer in $\{1, \dots, n\}$. By induction hypothesis, the two following conditions are equivalent:

1. $\forall g(s_1, \dots, s_l) \preceq t_j, \forall p \leq l, \text{root}(s_p) \in \text{Follow}(E, g, p),$
2. $\forall 1 \leq k \leq n_j, g_j^k \in \Delta^*(r_{j,k}).$

Hence:

$$\begin{aligned}
& \forall g(s_1, \dots, s_l) \preceq t, \forall p \leq l, \text{root}(s_p) \in \text{Follow}(E, g_i, p) \\
& \Leftrightarrow \forall 1 \leq j \leq m, \forall g(s_1, \dots, s_l) \preceq t_j, \forall p \leq l, \text{root}(s_p) \in \text{Follow}(E, g, p) \\
& \quad \wedge g_j = \text{root}(t_j) \in \text{Follow}(E, f, j) \\
& \Leftrightarrow \forall 1 \leq j \leq m, \forall 1 \leq k \leq n_j, g_j^k \in \Delta^*(s_k) \wedge (f^j, g_j, g_j^1, \dots, g_j^{l_j}) \in \Delta \\
& \Leftrightarrow \forall 1 \leq k \leq m, f^k \in \Delta^*(t_k).
\end{aligned}$$

As a direct consequence of the two previous propositions, it can be shown that the K -position automaton of E recognizes the language denoted by E .

Theorem 1. *Let E be linear. Then, $\mathcal{L}(\mathcal{P}_E) = \llbracket E \rrbracket$.*

Proof. Let t be a tree in T_Σ .

If $t = a \in \Sigma_0$, $a \in \mathcal{L}(\mathcal{P}_E) \Leftrightarrow a \in \text{First}(E) \Leftrightarrow a \in \llbracket E \rrbracket$.

If $t = f(t_1, \dots, t_n)$:

$$\begin{aligned}
t \in \mathcal{L}(\mathcal{P}_E) & \Leftrightarrow \varepsilon^1 \in \Delta^*(t) \\
& \Leftrightarrow (\varepsilon^1, f, f^1, \dots, f^n) \in \Delta \wedge \forall 1 \leq j \leq n, f_j \in \Delta^*(t_j) \\
& \Leftrightarrow f \in \text{First}(E) \wedge \forall g(s_1, \dots, s_l) \preceq t, \forall p \leq l, \text{root}(s_p) \in \text{Follow}(E, g, p) \quad (\text{Proposition 2}) \\
& \Leftrightarrow t \in \llbracket E \rrbracket \quad (\text{Proposition 1})
\end{aligned}$$

This construction can be extended to non-necessarily linear regular expression using the linearization and the mapping h . The K -Position Automaton \mathcal{P}_E associated with E is obtained by replacing each transition $(f_j^k, g_i, g_i^1, \dots, g_i^n)$ of the tree automaton $\mathcal{P}_{\bar{E}}$ by $(f_j^k, h(g_i), g_i^1, \dots, g_i^n)$.

Corollary 1. $h(\llbracket \bar{E} \rrbracket) = h(\mathcal{L}(\mathcal{P}_{\bar{E}})) = \mathcal{L}(\mathcal{P}_E) = \llbracket E \rrbracket$.

Example 2. Let $E = (f(a)^{*a} \cdot_a b + h(b))^{*b} + g(c, a)^{*c} \cdot_c (f(a)^{*a} \cdot_a b + h(b))^{*b}$ be the regular expression of the Example 1. The k -Position Automaton $\mathcal{P}_{\bar{E}}$ associated with \bar{E} is given in Figure 1. The set of states is $Q = \{\varepsilon^1, f_1^1, h_2^1, g_3^1, g_3^2, f_4^1, h_5^1\}$. The set of final states is $Q_T = \{\varepsilon^1\}$. The set of transition rules Δ is

$f_1(f_1^1) \rightarrow f_1^1$	$f_1(f_1^1) \rightarrow \varepsilon^1$	$f_1(h_2^1) \rightarrow \varepsilon^1$	$f_1(h_2^1) \rightarrow f_1^1$
$h_2(f_1^1) \rightarrow \varepsilon^1$	$b \rightarrow f_1^1$	$b \rightarrow h_2^1$	$h_2(f_1^1) \rightarrow h_2^1$
$h_2(h_2^1) \rightarrow \varepsilon^1$	$h_2(h_2^1) \rightarrow h_2^1$	$g_3(f_4^1, g_3^2) \rightarrow \varepsilon^1$	$b \rightarrow g_3^1$
$a \rightarrow g_3^2$	$g_3(h_5^1, g_3^2) \rightarrow \varepsilon^1$	$g_3(g_3^1, g_3^2) \rightarrow \varepsilon^1$	$f_4(f_4^1) \rightarrow f_4^1$
$f_4(h_5^1) \rightarrow \varepsilon^1$	$f_4(h_5^1) \rightarrow f_4^1$	$b \rightarrow f_4^1$	$f_4(f_4^1) \rightarrow \varepsilon^1$
$b \rightarrow h_5^1$	$h_5(f_4^1) \rightarrow \varepsilon^1$	$h_5(h_5^1) \rightarrow \varepsilon^1$	$h_5(f_4^1) \rightarrow h_5^1$
$h_5(h_5^1) \rightarrow h_5^1$	$b \rightarrow \varepsilon^1$		

The number of states is $|Q| = 7$ and the number of transition rules is $|\Delta| = 26$.

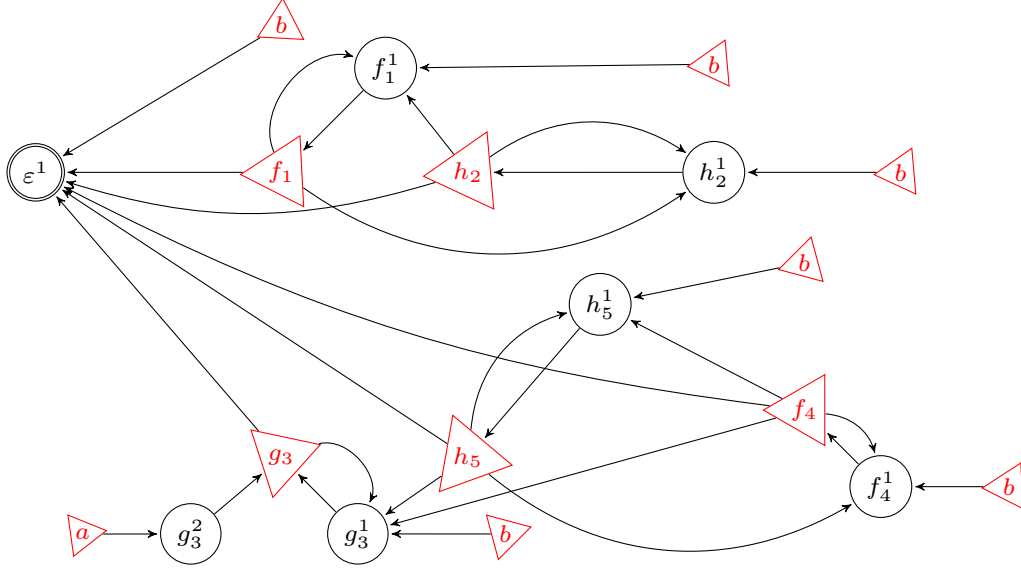


Fig. 1. The k -Position Automaton $\mathcal{P}_{\overline{\mathbf{E}}}$.

3.2 The Follow Tree Automaton

In this section, we define the follow tree automaton which is a generalisation of the Follow automaton introduced by L. Ilie and S. Yu in [9] in the case of words, and that it is a quotient of the K -position automaton, similarly to the case of word. Notice that in this automaton, states are no longer positions, but set of positions.

Definition 2. Let E be linear. The Follow Automaton of E is the tree automaton $\mathcal{F}_E = (Q, \Sigma, Q_T, \Delta)$ defined as follows

$$Q = \{\text{First}(E)\} \cup \bigcup_{f \in \Sigma_{E_m}} \{\text{Follow}(E, f, k) \mid 1 \leq k \leq m\}, \quad Q_T = \{\text{First}(E)\},$$

$$\Delta = \begin{cases} \{(\text{Follow}(E, g, l), f, \text{Follow}(E, f, 1), \dots, \text{Follow}(E, f, m)) \mid f \in \Sigma_{E_m} \wedge f \in \text{Follow}(E, g, l)\} \cup \\ \{(I, c) \mid c \in I \wedge c \in \Sigma_0\} \end{cases}$$

Let us show that \mathcal{F}_E is a quotient of \mathcal{P}_E w.r.t. a similarity relation ; since this kind of quotient preserves the language, this method is consequently a prove of the fact that the language denoted by E is recognized by \mathcal{F}_E .

Definition 3. A similarity relation over an automaton $A(Q, \Sigma, Q_T, \Delta)$ is a relation \sim over Q such that for any two states q and q' in Q :

$$q \sim q' \Rightarrow \forall f \in \Sigma_n, \forall (q_1, \dots, q_n) \in Q^n, (q, f, q_1, \dots, q_n) \in \Delta \Leftrightarrow (q', f, q_1, \dots, q_n) \in \Delta$$

Proposition 3. *Let \mathcal{A} be an automaton and \sim be a similarity relation over \mathcal{A} . Then $\mathcal{L}(\mathcal{A}_{/\sim}) = \mathcal{L}(\mathcal{A})$.*

Proof. Let us set $\mathcal{A} = (Q, \Sigma, Q_T, \Delta)$ and $\mathcal{A}_{/\sim} = (Q', \Sigma, Q'_T, \Delta')$. Let t be tree in T_Σ . Let us show by induction over the structure of t that $q \in \Delta^*(t) \Leftrightarrow [q] \in \Delta'^*(t)$.

1. Consider that $t = a \in \Sigma_0$. Then By construction of $\mathcal{A}_{/\sim}$, $(q, a) \in \Delta \Leftrightarrow ([q], a) \in \Delta'$.
2. Let us consider that $t = f(t_1, \dots, t_n)$ with f in Σ_n and t_1, \dots, t_n any n trees in T_Σ . Then:

$$\begin{aligned}
& q \in \Delta^*(t) \\
& \Leftrightarrow q \in \Delta^*(f(t_1, \dots, t_n)) \\
& \Leftrightarrow q \in \Delta(f, q_1, \dots, q_n) \text{ with } q_j \in \Delta^*(q_j) \text{ and } 1 \leq j \leq n \\
& \Leftrightarrow q \in \Delta(f, q_1, \dots, q_n) \text{ with } [q_j] \in \Delta'^*([q_j]) \text{ and } 1 \leq j \leq n \quad (\text{Induction hypothesis}) \\
& \Leftrightarrow q \in \Delta'(f, [q_1], \dots, [q_n]) \text{ with } [q_j] \in \Delta'^*([q_j]) \text{ and } 1 \leq j \leq n \quad (\text{Construction of } \mathcal{A}_{/\sim}) \\
& \Leftrightarrow [q] \in \Delta'^*(f(t_1, \dots, t_n)) \\
& \Leftrightarrow [q] \in \Delta'^*(t)
\end{aligned}$$

The quotient from \mathcal{P}_E to \mathcal{F}_E is defined by the following similarity relation. Notice that we extend the definition of the function Follow to the position ε^1 by $\text{Follow}(E, \varepsilon^1, 1) = \text{First}(E)$. Let E be linear and $\mathcal{P}_E = (Q, \Sigma, Q_T, \Delta)$. The *Follow Relation* is the relation $\sim_{\mathcal{F}}$ defined for any two states f^k and g^l in Q by $f^k \sim_{\mathcal{F}} g^l \Leftrightarrow \text{Follow}(E, f, k) = \text{Follow}(E, g, l)$.

Proposition 4. *Let E be linear. The relation $\sim_{\mathcal{F}}$ is the largest similarity relation over \mathcal{P}_E .*

Proof. Let us set $\mathcal{P}_E = (Q, \Sigma, Q_T, \Delta)$. Let f^k and g^l be two states in Q . Then:

$$\begin{aligned}
& f^k \sim_{\mathcal{F}} g^l \\
& \Leftrightarrow \text{Follow}(E, f, k) = \text{Follow}(E, g, l) \\
& \Leftrightarrow \forall h \in \Sigma_m, (h \in \text{Follow}(E, f, k) \Leftrightarrow h \in \text{Follow}(E, g, l)) \\
& \Leftrightarrow \forall h \in \Sigma_m, ((f^k, h, h^1, \dots, h^m) \in \Delta \Leftrightarrow (g^l, h, h^1, \dots, h^m) \in \Delta) \quad (\text{Construction of } \mathcal{P}_E) \\
& \Leftrightarrow \forall h \in \Sigma_m, \forall (q_1, \dots, q_m) \in Q^m, (f^k, h, q_1, \dots, q_m) \in \Delta \Leftrightarrow (g^l, h, q_1, \dots, q_m) \in \Delta \quad (\text{Construction of } \mathcal{P}_E)
\end{aligned}$$

Proposition 5. *Let E be linear. The finite tree automaton $\mathcal{P}_E / \sim_{\mathcal{F}}$ is isomorphic to \mathcal{F}_E .*

Proof. Let us set $\mathcal{P}_E / \sim_{\mathcal{F}} = (Q, \Sigma, Q_T, \Delta)$ and $\mathcal{F}_E = (Q', \Sigma, Q'_T, \Delta')$. Let us consider the mapping ϕ defined from Q to Q' by $\phi([f^k]) = \text{Follow}(E, f, k)$ ($\phi([\varepsilon^1]) = \text{Follow}(E, \varepsilon^1, 1) = \text{First}(E)$). Let us show that any tuple $([f^k], g, [g^1], \dots, [g^m])$ is a transition in Δ if and only if the tuple $(\phi([f^k]), g, \phi([g^1]), \dots, \phi([g^m]))$ is a transition in Δ' . Let f^k and g^l be two states in Q . Then:

$$\begin{aligned}
& ([f^k], g, [g^1], \dots, [g^m]) \in \Delta \\
& \Leftrightarrow g \in \text{Follow}(E, f, k) \quad (\text{Definition of } \sim_{\mathcal{F}}) \\
& \Leftrightarrow (\text{Follow}(E, f, k), g, \text{Follow}(E, g, 1), \dots, \text{Follow}(E, g, m)) \in \Delta \quad (\text{Definition of } \mathcal{F}_E) \\
& \Leftrightarrow (\phi([f^k]), g, \phi([g^1]), \dots, \phi([g^m])) \in \Delta'
\end{aligned}$$

As a direct consequence of the previous results, the following theorem can be shown.

Theorem 2. *Let E be linear. Then $\mathcal{L}(\mathcal{F}_E) = \llbracket E \rrbracket$.*

Proof. According to Theorem 1, $\llbracket E \rrbracket = \mathcal{L}(\mathcal{P}_E)$. According to Proposition 5, \mathcal{F}_E is a quotient of \mathcal{P}_E w.r.t. $\sim_{\mathcal{F}}$, which is a similarity relation following Proposition 4. Consequently, Proposition 3 asserts that $\mathcal{L}(\mathcal{P}_E) = \mathcal{L}(\mathcal{F}_E)$. Therefore $\mathcal{L}(\mathcal{F}_E) = \llbracket E \rrbracket$.

Finally, this method can be extended to non-necessarily linear regular expression as follows. The *Follow Automaton* \mathcal{F}_E associated with E is obtained by replacing each transition $(I, f_j, \text{Follow}(E, f_j, 1), \dots, \text{Follow}(E, f_j, m))$ of \mathcal{F}_E by $(I, h(f_j), \text{Follow}(E, f_j, 1), \dots, \text{Follow}(E, f_j, m))$.

Corollary 2. $\mathcal{L}(\mathcal{F}_E) = \llbracket E \rrbracket$.

Example 3. The Follow Automaton \mathcal{F}_E associated with $E = (f(a)^{*a} \cdot_a b + h(b))^{*b} + g(c, a)^{*c} \cdot_c (f(a)^{*a} \cdot_a b + h(b))^{*b}$ of the Example 1 is given in Figure 2.

The set of states is $Q = \{\{a\}, \{b, f_1, h_2\}, \{b, f_1, h_2, g_3, f_4, h_5\}, \{b, g_3, f_4, h_5\}, \{b, f_4, h_5\}\}$ and

$Q_T = \{\{b, f_1, h_2, g_3, f_4, h_5\}\}$. The set of transition rules Δ is

$$\begin{array}{lll}
b \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} & b \rightarrow \{b, g_3, f_4, h_5\} & a \rightarrow \{a\} \\
g(\{b, g_3, f_4, h_5\}, \{a\}) \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} & b \rightarrow \{b, f_1, h_2\} & \\
g(\{b, g_3, f_4, h_5\}, \{a\}) \rightarrow \{b, g_3, f_4, h_5\} & b \rightarrow \{b, f_4, h_5\} & \\
f(\{b, f_1, h_2\}) \rightarrow \{b, f_1, h_2\} & h(\{b, f_1, h_2\}) \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} & \\
h(\{b, f_1, h_2\}) \rightarrow \{b, f_1, h_2\} & f(\{b, f_4, h_5\}) \rightarrow \{b, f_4, h_5\} & \\
f(\{b, f_4, h_5\}) \rightarrow \{b, g_3, f_4, h_5\} & f(\{b, f_4, h_5\}) \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} & \\
h(\{b, f_4, h_5\}) \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} & h(\{b, f_4, h_5\}) \rightarrow \{b, f_4, h_5\} & \\
h(\{b, f_4, h_5\}) \rightarrow \{b, g_3, f_4, h_5\} & f(\{b, f_1, h_2\}) \rightarrow \{b, f_1, h_2, g_3, f_4, h_5\} &
\end{array}$$

The number of states is $|Q| = 5$ and the number of transition rules is $|\Delta| = 17$.

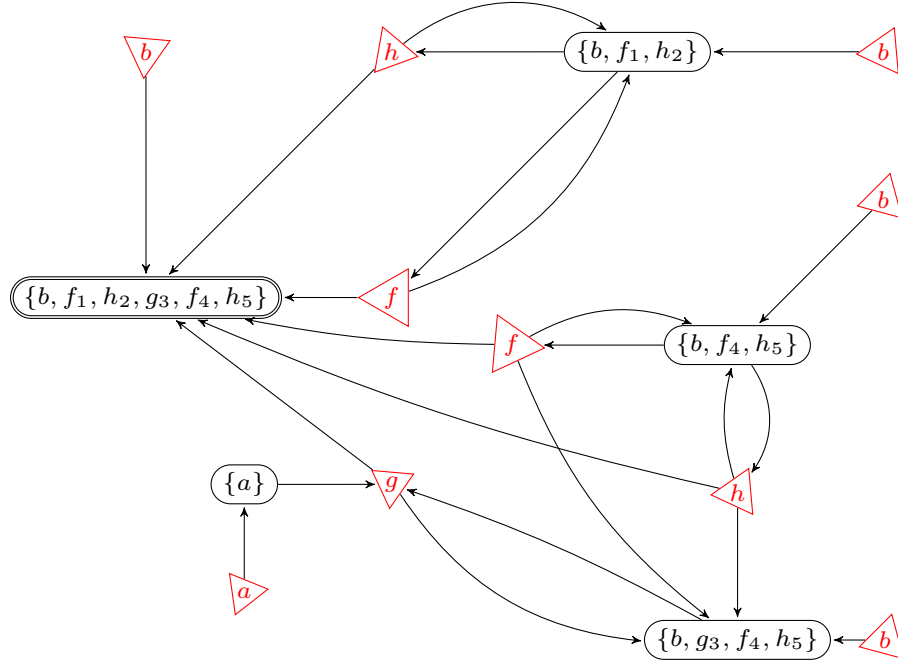


Fig. 2. The Follow Automaton \mathcal{F}_E .

3.3 The Equation Tree Automaton

In [11], Kuske and Meinecke extend the notion of word partial derivatives [1] to tree partial derivatives in order to compute from E a tree automaton recognizing $\llbracket E \rrbracket$. Due to the notion of ranked alphabet, partial derivatives are no longer sets of expressions, but sets of tuples of expressions.

Let $\mathcal{N} = (E_1, \dots, E_n)$ be a tuple of regular expressions, F be some regular expression and $c \in \Sigma_0$. Then $\mathcal{N} \cdot_c F$ is the tuple $(E_1 \cdot_c F, \dots, E_n \cdot_c F)$. For \mathcal{S} a set of tuples of regular expressions, $\mathcal{S} \cdot_c F$ is the set $\mathcal{S} \cdot_c F = \{\mathcal{N} \cdot_c F \mid \mathcal{N} \in \mathcal{S}\}$. Finally, $\text{SET}(\mathcal{N}) = \{E_1, \dots, E_m\}$ and $\text{SET}(\mathcal{S}) = \bigcup_{\mathcal{N} \in \mathcal{S}} \text{SET}(\mathcal{N})$.

Let f be a symbol in $\Sigma_{>0}$. The set $f^{-1}(E)$ of tuples of regular expressions is defined as follows:

$$\begin{aligned}
f^{-1}(0) &= \emptyset, \\
f^{-1}(g(E_1, \dots, E_n)) &= \begin{cases} \{(E_1, \dots, E_n)\} & \text{if } f = g, \\ \emptyset & \text{otherwise,} \end{cases} \\
f^{-1}(E_1 + E_2) &= f^{-1}(E_1) \cup f^{-1}(E_2), \\
f^{-1}(E_1^{*c}) &= f^{-1}(E_1) \cdot_c E_1^{*c}, \\
f^{-1}(E_1 \cdot_c E_2) &= \begin{cases} f^{-1}(E_1) \cdot_c E_2 & \text{if } c \notin \llbracket E_1 \rrbracket \\ f^{-1}(E_1) \cdot_c E_2 \cup f^{-1}(E_2) & \text{otherwise.} \end{cases}
\end{aligned}$$

The function f^{-1} is extended to any set S of regular expressions by $f^{-1}(S) = \bigcup_{E \in S} f^{-1}(E)$.

The *partial derivative* of E w.r.t. a word $w \in \Sigma_{\geq 1}^*$, denoted by $\partial_w(E)$, is the set of regular expressions inductively defined by:

$$\partial_w(E) = \begin{cases} \{E\} & \text{if } w = \varepsilon, \\ \text{SET}(f^{-1}(\partial_u(E))) & \text{if } w = uf, f \in \Sigma_{\geq 1}, u \in \Sigma_{\geq 1}^*, f^{-1}(\partial_u(E)) \neq \emptyset, \\ \{0\} & \text{if } f^{-1}(\partial_u(E)) = \emptyset. \end{cases}$$

The *Equation Automaton* of E is the tree automaton $\mathcal{A}_E = (Q, \Sigma, Q_T, \Delta)$ defined by $Q = \{\partial_w(E) \mid w \in \Sigma_{\geq 1}^*\}$, $Q_T = \{E\}$, and

$$\Delta = \left\{ \begin{aligned} & \{(F, f, G_1, \dots, G_m) \mid F \in Q, f \in \Sigma_m, m \geq 1, (G_1, \dots, G_m) \in f^{-1}(F)\} \\ & \cup \{(F, c) \mid c \in (\llbracket F \rrbracket \cap \Sigma_0)\} \end{aligned} \right\}$$

Example 4. Let $E = \underbrace{(f(a)^*a \cdot_a b + h(b))^*b}_F + \underbrace{g(c, a)^*c}_G \cdot_c \underbrace{(f(a)^*a \cdot_a b + h(b))^*b}_F$ (Example 1).

$$\begin{aligned} \partial_f(E) &= \{((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F\} & \partial_g(E) &= \{(a \cdot_c G) \cdot_c F, (c \cdot_c G) \cdot_c F\} & \partial_{fh}(E) &= \{b \cdot_b F\} \\ \partial_{ff}(E) &= \{((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F\} & \partial_{gf}(E) &= \{((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F\} & \partial_{gh}(E) &= \{b \cdot_b F\} \\ \partial_{hf}(E) &= \{((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F\} & \partial_{hh}(E) &= \{((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F\} & \partial_h(E) &= \{b \cdot_b F\} \\ \partial_{gg}(E) &= \{(a \cdot_c G) \cdot_c F, (c \cdot_c G) \cdot_c F\} \end{aligned}$$

The set of states Q is $q_0 = E$, $q_1 = ((a \cdot_a f(a)^*a) \cdot_a b) \cdot_b F$, $q_2 = b \cdot_b F$, $q_3 = (c \cdot_c G) \cdot_c F$, $q_4 = (a \cdot_c G) \cdot_c F$.

The set of final states is $Q_T = \{q_0\}$. The set of transition rules is

$$\begin{array}{lllll} b \rightarrow q_0 & b \rightarrow q_1 & b \rightarrow q_3 & b \rightarrow q_2 & f(q_1) \rightarrow q_0 \\ h(q_2) \rightarrow q_0 & g(q_3, q_4) \rightarrow q_0 & h(q_2) \rightarrow q_1 & g(q_3, q_4) \rightarrow q_4 & f(q_1) \rightarrow q_1 \\ h(q_2) \rightarrow q_2 & f(q_1) \rightarrow q_4 & h(q_2) \rightarrow q_4 & a \rightarrow q_4 & \end{array}$$

The number of states is $|Q| = 5$ and the number of transition rules is $|\Delta| = 15$. The Equation Automaton associated with E is given in Figure 3.

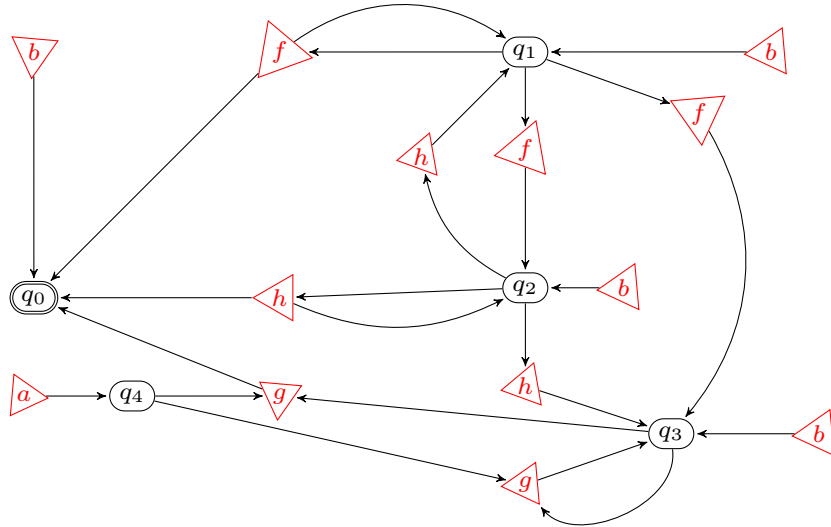


Fig. 3. The Equation Automaton \mathcal{A}_E .

3.4 The k -C-Continuation Tree Automaton

In [11], Kuske and Meinecke show how to efficiently compute the equation tree automaton of a regular expression *via* an extension of Champarnaud and Ziadi's C-Continuation [3, 4, 10]. In [13, 14], we show how

to inductively compute them. We also show how to efficiently compute the k -C-Continuation tree automaton associated with a regular expression. In this section, we prove that this automaton is isomorphic to the k -position tree automaton, similarly to the case of words. Recall that the following equations are defined, as already, modulo the trivial identities.

Definition 4 ([13, 14]). Let $E \neq 0$ be linear. Let k and m be two integers such that $1 \leq k \leq m$. Let f be in $(\Sigma_E \cap \Sigma_m)$. The k -C-continuation $C_{f^k}(E)$ of f in E is the regular expression defined by:

$$\begin{aligned} C_{f^k}(g(E_1, \dots, E_m)) &= \begin{cases} E_k & \text{if } f = g \\ C_{f^k}(E_j) & \text{if } f \in \Sigma_{E_j} \end{cases} \\ C_{f^k}(E_1 + E_2) &= \begin{cases} C_{f^k}(F) & \text{if } f \in \Sigma_F \\ C_{f^k}(G) & \text{if } f \in \Sigma_G \end{cases} \\ C_{f^k}(F \cdot_c G) &= \begin{cases} C_{f^k}(F) \cdot_c G & \text{if } f \in \Sigma_F \\ C_{f^k}(G) & \text{if } f \in \Sigma_G \\ & \text{and } c \in \text{Last}(F) \\ 0 & \text{otherwise} \end{cases} \\ C_{f^k}(F^{*c}) &= C_{f^k}(F) \cdot_c F^{*c} \end{aligned}$$

By convention, we set $C_{\varepsilon^1}(E) = E$.

Let us now show how to compute the k -C-Continuation tree automaton.

Definition 5 ([13, 14]). Let $E \neq 0$ be linear. The automaton $\mathcal{C}_E = (Q_C, \Sigma_E, \{C_{\varepsilon^1}(E)\}, \Delta_C)$ is defined by

$$\begin{aligned} - Q_C &= \{(f^k, C_{f^k}(E)) \mid f \in \Sigma_m, 1 \leq k \leq m\} \cup \{(\varepsilon^1, C_{\varepsilon^1}(E))\}, \\ - \Delta_C &= \left\{ \begin{aligned} &\{((x, C_x(E)), g, ((g^1, C_{g^1}(E)), \dots, (g^m, C_{g^m}(E)))) \mid g \in \Sigma_{E_m}, \\ &m \geq 1, (C_{g^1}(E), \dots, C_{g^m}(E)) \in g^{-1}(C_x(E))\} \\ &\cup \{((x, C_x(E)), c) \mid c \in \llbracket C_x(E) \rrbracket \cap \Sigma_0\} \end{aligned} \right\} \end{aligned}$$

The C -Continuation tree automaton \mathcal{C}_E associated with E is obtained by relabelling the transitions of \mathcal{C}_E using the mapping h .

Theorem 3 ([13, 14]). The automaton \mathcal{C}_E accepts $\llbracket E \rrbracket$.

Example 5. Let $E = \underbrace{(f(a)^{*a} \cdot_a b + h(b))^{*b}}_{F_1} + \underbrace{g(c, a)^{*c} \cdot_c (f(a)^{*a} \cdot_a b + h(b))^{*b}}_{G_2}$ defined in Example 1 and $\overline{E} = \underbrace{(f_1(a)^{*a} \cdot_a b + h_2(b))^{*b}}_{F_1} + \underbrace{g_3(c, a)^{*c} \cdot_c (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b}}_{F_3}$.

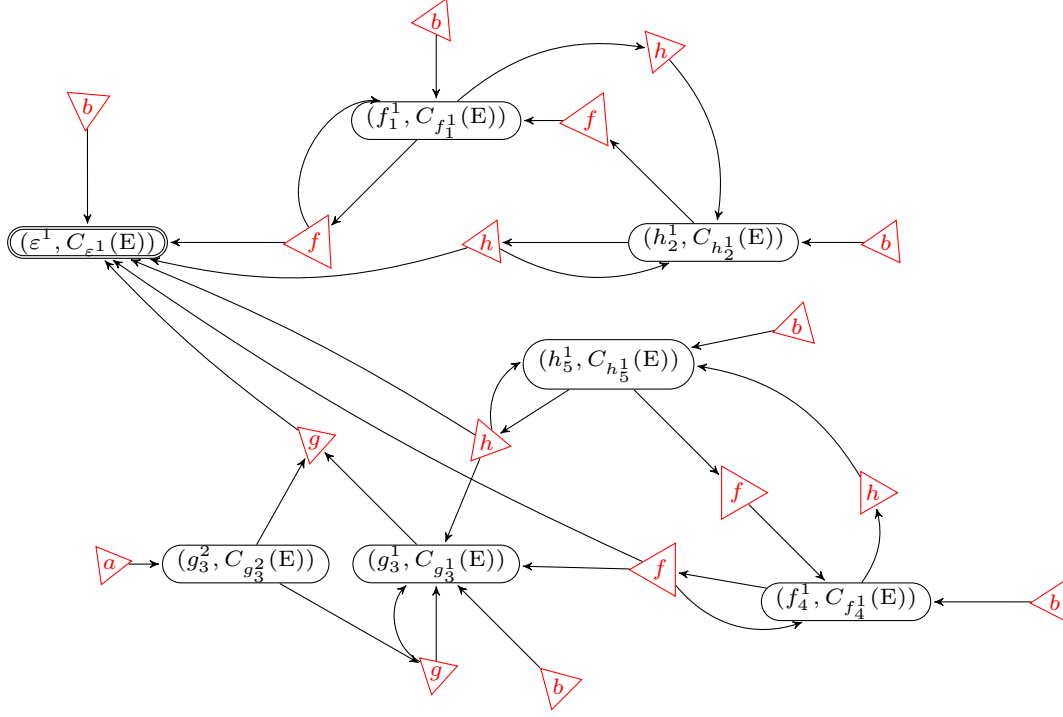


Fig. 4. The k -C-Continuation Automaton \mathcal{C}_E .

The computation of the k -C-Continuations of E using the Definition 4 is given in Table 1.

$C_{f_1^1}(\bar{E}) = ((a \cdot_a f_1(a)^{*a}) \cdot_a b) \cdot_b F_1$	$h(C_{f_1^1}(\bar{E})) = ((a \cdot_a f(a)^{*a}) \cdot_a b) \cdot_b F$
$C_{h_2^1}(\bar{E}) = b \cdot_b F_1$	$h(C_{h_2^1}(\bar{E})) = b \cdot_b F$
$C_{g_3^1}(\bar{E}) = (c \cdot_c g_3(c, a)^{*c}) \cdot_c F_3$	$h(C_{g_3^1}(\bar{E})) = (c \cdot_c g(c, a)^{*c}) \cdot_c F$
$C_{g_3^2}(\bar{E}) = (a \cdot_c g_3(c, a)^{*c}) \cdot_c F_3$	$h(C_{g_3^2}(\bar{E})) = (a \cdot_c g(c, a)^{*c}) \cdot_c F$
$C_{f_4^1}(\bar{E}) = ((a \cdot_a f_4(a)^{*a}) \cdot_a b) \cdot_b F_3$	$h(C_{f_4^1}(\bar{E})) = ((a \cdot_a f(a)^{*a}) \cdot_a b) \cdot_b F$
$C_{h_5^1}(\bar{E}) = b \cdot_b F_3$	$h(C_{h_5^1}(\bar{E})) = b \cdot_b F$

Table 1. The k -C-Continuations of \bar{E} .

The set of states of the automaton \mathcal{C}_E is $Q = \{(\varepsilon^1, C_{\varepsilon^1}(\bar{E})), (f_1^1, C_{f_1^1}(\bar{E})), (h_2^1, C_{h_2^1}(\bar{E})), (g_3^1, C_{g_3^1}(\bar{E})), (g_3^2, C_{g_3^2}(\bar{E})), (f_4^1, C_{f_4^1}(\bar{E})), (h_5^1, C_{h_5^1}(\bar{E}))\}$.

The set of transition rules Δ is

$$\begin{array}{ll}
b \rightarrow (f_1^1, C_{f_1^1}(\bar{E})) & b \rightarrow (h_2^1, C_{h_2^1}(\bar{E})) \\
a \rightarrow (g_3^2, C_{g_3^2}(\bar{E})) & b \rightarrow (h_5^1, C_{h_5^1}(\bar{E})) \\
b \rightarrow (g_3^1, C_{g_3^1}(\bar{E})) & b \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E})) \quad b \rightarrow (f_4^1, C_{f_4^1}(\bar{E})) \\
f((f_1^1, C_{f_1^1}(\bar{E}))) \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E})) & f((f_4^1, C_{f_4^1}(\bar{E}))) \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E})) \\
h((h_5^1, C_{h_5^1}(\bar{E}))) \rightarrow (g_3^1, C_{g_3^1}(\bar{E})) & h((h_5^1, C_{h_5^1}(\bar{E}))) \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E})) \\
f((f_1^1, C_{f_1^1}(\bar{E}))) \rightarrow (f_1^1, C_{f_1^1}(\bar{E})) & h((h_2^1, C_{h_2^1}(\bar{E}))) \rightarrow (h_2^1, C_{h_2^1}(\bar{E})) \\
h((h_2^1, C_{h_2^1}(\bar{E}))) \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E})) & f((f_1^1, C_{f_1^1}(\bar{E}))) \rightarrow (h_2^1, C_{h_2^1}(\bar{E})) \\
f((f_4^1, C_{f_4^1}(\bar{E}))) \rightarrow (f_4^1, C_{f_4^1}(\bar{E})) & f((f_4^1, C_{f_4^1}(\bar{E}))) \rightarrow (h_5^1, C_{h_5^1}(\bar{E})) \\
h((h_2^1, C_{h_2^1}(\bar{E}))) \rightarrow (f_1^1, C_{f_1^1}(\bar{E})) & h((h_5^1, C_{h_5^1}(\bar{E}))) \rightarrow (h_5^1, C_{h_5^1}(\bar{E})) \\
f((f_4^1, C_{f_4^1}(\bar{E}))) \rightarrow (g_3^1, C_{g_3^1}(\bar{E})) & h((h_5^1, C_{h_5^1}(\bar{E}))) \rightarrow (f_4^1, C_{f_4^1}(\bar{E})) \\
g((g_3^1, C_{g_3^1}(\bar{E})), (g_3^2, C_{g_3^2}(\bar{E}))) \rightarrow (g_3^1, C_{g_3^1}(\bar{E})) & g((g_3^1, C_{g_3^1}(\bar{E})), (g_3^2, C_{g_3^2}(\bar{E}))) \rightarrow (\varepsilon^1, C_{\varepsilon^1}(\bar{E}))
\end{array}$$

The number of states is $|Q| = 5$ and the number of transition rules is $|\Delta| = 15$. The k -C-Continuation Automaton associated with E is given in Figure 4.

Let \sim_e be the equivalence relation over the set of states of \mathcal{C}_E defined for any two states $(f_j^k, C_{f_j^k}(\bar{E}))$ and $(g_i^p, C_{g_i^p}(\bar{E}))$ by $(f_j^k, C_{f_j^k}(\bar{E})) \sim_e (g_i^p, C_{g_i^p}(\bar{E})) \Leftrightarrow h(C_{f_j^k}(\bar{E})) = h(C_{g_i^p}(\bar{E}))$.

Proposition 6 ([13, 14]). *The automaton \mathcal{C}_E / \sim_e is isomorphic to \mathcal{A}_E .*

Example 6. Using the equivalence-relation \sim_e over the set of states of k -C-Continuation Automaton \mathcal{C}_E (Figure 4) we see that $h(C_{f_1^1}(\bar{E})) = h(C_{f_4^1}(\bar{E}))$ and $h(C_{h_2^1}(\bar{E})) = h(C_{h_5^1}(\bar{E}))$. The automaton \mathcal{C}_E / \sim_e is given in Figure 5.

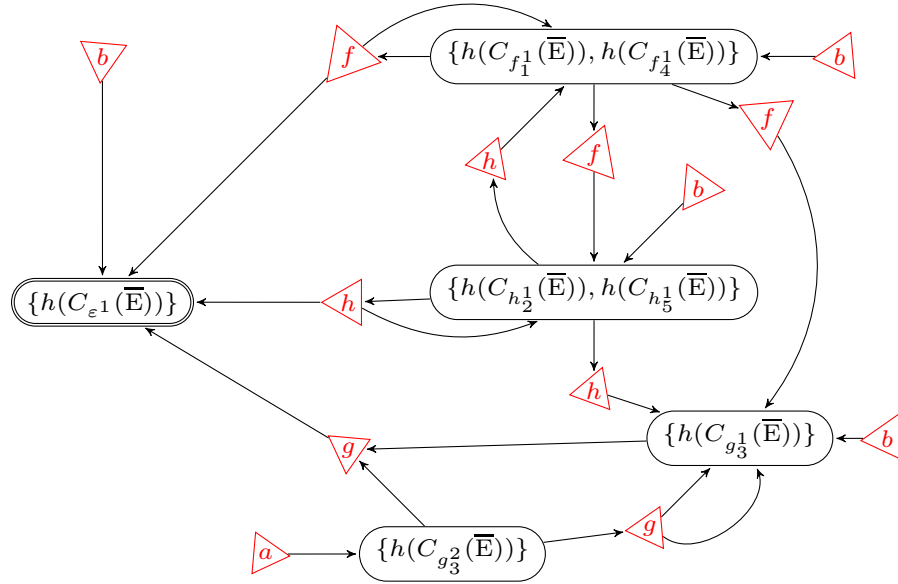


Fig. 5. The Automaton \mathcal{C}_E / \sim_e .

In order to show that the k -C-continuation tree automaton of E is isomorphic to the k -position automaton of E , we first show the link between the position functions and the C-continuations.

Proposition 7 ([13, 14]). *Let E be linear, $1 \leq k \leq m$ be two integers and f be a position in Σ_{E_m} . Then $\text{Follow}(E, f, k) = \text{First}(C_{f^k}(\bar{E}))$.*

Lemma 3. *Let E be linear and g be a symbol in $\Sigma_{\geq 1}$. Then $g^{-1}(E) \neq \emptyset \Leftrightarrow g \in \text{First}(E)$.*

Proof. By induction on the construction of E .

1. If $E = a$, then $g^{-1}(E) = \emptyset$ and $g \notin \text{First}(E) = \{a\}$.

2. Suppose that $E = f(E_1, \dots, E_n)$. Hence $\text{First}(E) = \{f\}$. Then:

$$\begin{aligned} g^{-1}(E) \neq \emptyset &\Leftrightarrow f = g \\ &\Leftrightarrow g \in \text{First}(E) \end{aligned}$$

3. Consider that $E = E_1 + E_2$. Then:

$$\begin{aligned} g^{-1}(E) \neq \emptyset &\Leftrightarrow g^{-1}(E_1) \neq \emptyset \vee g^{-1}(E_2) \neq \emptyset \\ &\Leftrightarrow g \in \text{First}(E_1) \vee g \in \text{First}(E_2) \quad (\text{Induction Hypothesis}) \\ &\Leftrightarrow g \in \text{First}(E) \end{aligned}$$

4. Suppose that $E = E_1 \cdot_c E_2$. Two cases have to be considered.

(a) Suppose that $c \in \llbracket E_1 \rrbracket$. Then:

$$\begin{aligned} g^{-1}(E) \neq \emptyset &\Leftrightarrow g^{-1}(E_1) \cdot_c E_2 \neq \emptyset \vee g^{-1}(E_2) \neq \emptyset \\ &\Leftrightarrow g^{-1}(E_1) \neq \emptyset \vee g^{-1}(E_2) \neq \emptyset \\ &\Leftrightarrow g \in \text{First}(E_1) \vee g \in \text{First}(E_2) \quad (\text{Induction Hypothesis}) \\ &\Leftrightarrow g \in \text{First}(E_1) \setminus \{c\} \vee g \in \text{First}(E_2) \quad (g \neq c) \\ &\Leftrightarrow g \in \text{First}(E) \end{aligned}$$

(b) Consider that $c \notin \llbracket E_1 \rrbracket$. Then:

$$\begin{aligned} g^{-1}(E) \neq \emptyset &\Leftrightarrow g^{-1}(E_1) \cdot_c E_2 \neq \emptyset \\ &\Leftrightarrow g^{-1}(E_1) \neq \emptyset \\ &\Leftrightarrow g \in \text{First}(E_1) \quad (\text{Induction Hypothesis}) \\ &\Leftrightarrow g \in \text{First}(E) \end{aligned}$$

5. Suppose that $E = E_1^{*c}$. Then:

$$\begin{aligned} g^{-1}(E) \neq \emptyset &\Leftrightarrow g^{-1}(E_1) \cdot_c E_1^{*c} \neq \emptyset \\ &\Leftrightarrow g^{-1}(E_1) \neq \emptyset \\ &\Leftrightarrow g \in \text{First}(E_1) \quad (\text{Induction Hypothesis}) \\ &\Leftrightarrow g \in \text{First}(E) \end{aligned}$$

Corollary 3. Let E be linear, $1 \leq k \leq m$ be two integers and f and g be two symbols in Σ . Then, $g^{-1}(C_{f^k}(E)) \neq \emptyset \Leftrightarrow g \in \text{First}(C_{f^k}(E))$.

Lemma 4. Let E be linear, $1 \leq k \leq m$ be two integers and f and g be two symbols in Σ . Then, $g^{-1}(C_{f^k}(E)) \neq \emptyset \Leftrightarrow g \in \text{Follow}(E, f, k)$.

Proof. According to Corollary 3, $g^{-1}(C_{f^k}(E)) \neq \emptyset \Leftrightarrow g \in \text{First}(C_{f^k}(E))$. Following Proposition 7, $\text{First}(C_{f^k}(E)) = \text{Follow}(F, f, k)$. Consequently, $g^{-1}(C_{f^k}(E)) \neq \emptyset \Leftrightarrow g \in \text{Follow}(F, f, k)$.

Proposition 8. Let E be linear. The automaton \mathcal{C}_E is isomorphic to \mathcal{P}_E .

Proof. Let $\mathcal{C}_E = (Q, \Sigma, Q_T, \Delta)$ and $\mathcal{P}_E = (Q', \Sigma, Q'_T, \Delta')$. Let ϕ be the bijection between Q and Q' defined for any state $q_{f^k} = (f^k, C_{f^k}(E))$ in Q by $\phi(q_{f^k}) = f_k$. Then for any symbol f in Σ_m :

$$\begin{aligned} &((g^l, C_{g^l}(E)), f, (f^1, C_{f^1}(E)), \dots, (f^m, C_{f^m}(E))) \in \Delta \\ &\Leftrightarrow (C_{f^1}(E), \dots, C_{f^m}(E)) \in f^{-1}(C_{g^l}(E)) \quad (\text{Definition of } \mathcal{C}_E) \\ &\Leftrightarrow f^{-1}(C_{g^l}(E)) \neq \emptyset \\ &\Leftrightarrow f \in \text{Follow}(E, g, l) \quad (\text{Lemma 4}) \\ &\Leftrightarrow (g^l, f, f^1, \dots, f^m) \in \Delta' \quad (\text{Construction of } \mathcal{P}_E) \\ &\Leftrightarrow (\phi((g^l, C_{g^l}(E))), f, \phi((f^1, C_{f^1}(E))), \dots, \phi((f^m, C_{f^m}(E)))) \in \Delta' \end{aligned}$$

This proposition can be extended to non-necessarily linear expression since \mathcal{C}_E and \mathcal{P}_E are relabelling of $\mathcal{C}_{\bar{E}}$ and $\mathcal{P}_{\bar{E}}$.

Corollary 4. The automaton \mathcal{C}_E is isomorphic to \mathcal{P}_E .

We define the equivalence relation denoted by \equiv over the set of states of the automaton \mathcal{C}_E as follows:

$$(f^k, C_{f^k}(\bar{E})) \equiv (g^p, C_{g^p}(\bar{E})) \Leftrightarrow \text{Follow}(\bar{E}, f, k) = \text{Follow}(\bar{E}, g, p).$$

Corollary 5. The finite tree automaton \mathcal{C}_E / \equiv is isomorphic to the follow automaton \mathcal{F}_E .

4 Comparison between the Equation and the Follow Automata

We discuss in this section two examples to compare the equation and the follow automata.

Let $\Sigma = \Sigma_0 \cup \Sigma_1$ be the ranked alphabet defined by $\Sigma_0 = \{a\}$ and $\Sigma_1 = \{f_1, \dots, f_n\}$. Let us consider the linear regular expression $E = ((f_1(a)^{*a} \cdot_a f_2(a)^{*a}) \cdot_a \dots) \cdot_a f_n(a)^{*a})^{*a}$ defined over Σ . Then the size of E is $|E| = 4n - 1$ and its alphabet width is $\|E\| = n + 1$. We have $\text{First}(E) = \{a, f_1, f_2, \dots, f_n\}$ and $\text{Follow}(E, f_1, 1) = \text{Follow}(E, f_2, 1) = \dots = \text{Follow}(E, f_n, 1) = \{a, f_1, f_2, \dots, f_n\}$.

The partial derivatives associated with E are:

$$\begin{aligned}\partial_{f_1}(E) &= \{(((a \cdot_a f_1(a)^{*a} \cdot_a f_2(a)^{*a}) \cdot_a \dots) \cdot_a f_n(a)^{*a}) \cdot_a E\} \\ \partial_{f_2}(E) &= \{((a \cdot_a f_2(a)^{*a} \cdot_a \dots) \cdot_a f_n(a)^{*a}) \cdot_a E\}, \dots \\ \partial_{f_n}(E) &= \{(a \cdot_a f_n(a)^{*a}) \cdot_a E\}.\end{aligned}$$

The K -position automaton associated with E has $n + 1$ states.

The follow automaton associated with E has 1 state.

The equation automaton associated with E has: $n + 1$ states.

Let $F = \underbrace{(f(a)^{*a} + f(a)^{*a} + \dots + f(a)^{*a})}_{f(a)^{*a} \text{ } n\text{-times}}$ be a regular expression defined over the ranked alphabet $\Sigma =$

$\Sigma_0 \cup \Sigma_1$ such that $\Sigma_0 = \{a\}$ and $\Sigma_1 = \{f\}$. We have $|E| = 4n - 1$ and $\|E\| = n + 1$. The linearized form associated with F is $\bar{F} = (f_1(a)^{*a} + f_2(a)^{*a} + \dots + f_n(a)^{*a})$. The set $\text{First}(F) = \{a, f_1, f_2, \dots, f_n\}$, $\text{Follow}(\bar{F}, f_1, 1) = \{a, f_1\}$, $\text{Follow}(\bar{F}, f_2, 1) = \{a, f_2\}, \dots$, and $\text{Follow}(\bar{F}, f_n, 1) = \{a, f_n\}$.

The partial derivatives associated with F are $\partial_f(F) = \{a \cdot_a f(a)^{*a}\}$, $\partial_{ff}(F) = \{a \cdot_a f(a)^{*a}\}$.

The K -position automaton associated with F has $n + 1$ states.

The follow automaton associated with F has: $n + 1$ states.

The equation automaton associated with F has: 2 states.

From these examples we state that the two automata are incomparable:

Proposition 9. *The follow tree automaton and the Equation Tree Automaton are incomparable though they derived from two isomorphic automata.*

4.1 A smaller automaton

In [7] P. García *et al.* proposed an algorithm to obtain an automaton from a word regular expression. Their method is based in the computation of both the partial derivatives automaton and the follow automaton. They join two relations, the first one is over the states of word follow automaton and the second one is over the word c-continuations automaton, in one relation denoted by \equiv_V . What we propose is to extend the relation \equiv_V to the case of trees as following:

$$C_{f_j^k}(\bar{E}) \equiv_V C_{g_i^p}(\bar{E}) \Leftrightarrow \begin{cases} (\exists C_{h_m^l}(\bar{E}) \sim_{\mathcal{F}} C_{f_j^k}(\bar{E}) \mid C_{h_m^l}(\bar{E}) \sim_e C_{g_i^p}(\bar{E})) \\ \vee (\exists C_{h_m^l}(\bar{E}) \sim_{\mathcal{F}} C_{g_i^p}(\bar{E}) \mid C_{h_m^l}(\bar{E}) \sim_e C_{f_j^k}(\bar{E})) \end{cases}$$

The idea is to define the follow relation $\sim_{\mathcal{F}}$ over the states of the c-continuation automaton \mathcal{C}_E as follows: $C_{f_j^k}(\bar{E}) \sim_{\mathcal{F}} C_{g_i^p}(\bar{E}) \Leftrightarrow \text{Follow}(C_{f_j^k}(\bar{E}), f_j, k) = \text{Follow}(C_{g_i^p}(\bar{E}), g_i, p)$ such that we keep all the equivalent k - \bar{C} -Continuations of the merged states. The obtained automaton is denoted by $\mathcal{C}_E / \sim_{\mathcal{F}}$. Then apply the equivalent relation \sim_e (apply the mapping h) over the states of the automaton $\mathcal{C}_E / \sim_{\mathcal{F}}$ and merge the states which have at least one expression in common.

Example 7. Let $E = (f(a)^{*a} \cdot_a b + h(b))^{*b} + g(c, a)^{*c} \cdot_c (f(a)^{*a} \cdot_a b + h(b))^{*b}$ defined in Example 1 and $\bar{E} = \underbrace{(f_1(a)^{*a} \cdot_a b + h_2(b))^{*b}}_{F_1} + \underbrace{g_3(c, a)^{*c} \cdot_c (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b}}_{F_3}$.

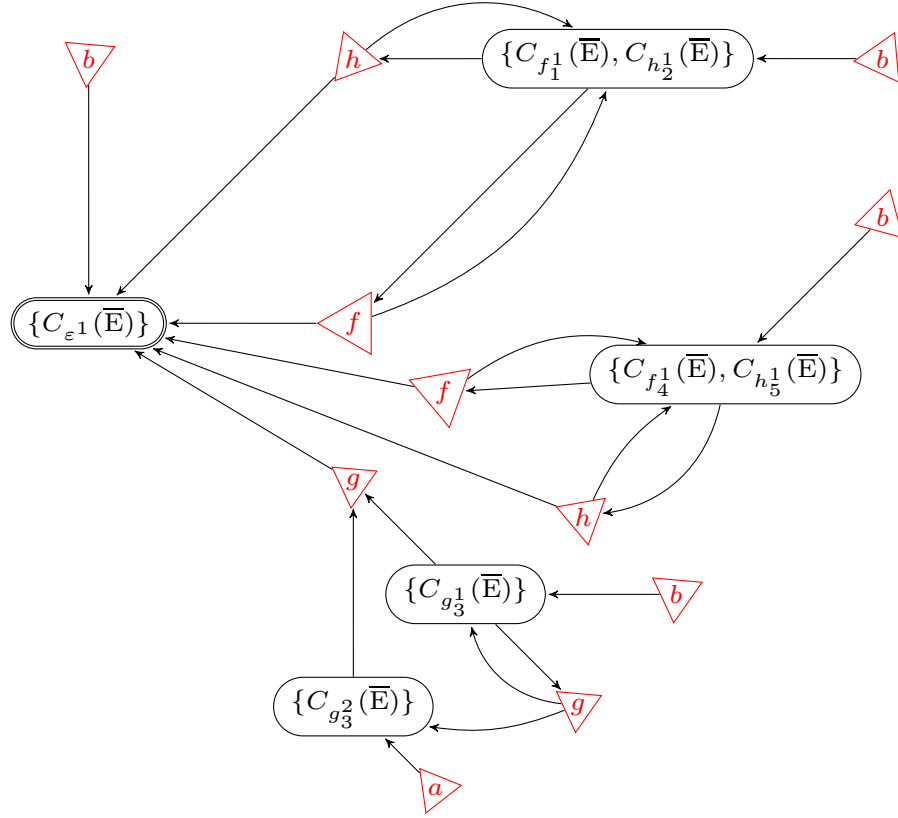


Fig. 6. The Automaton $\mathcal{C}_E / \sim_{\mathcal{F}}$.

$$\begin{aligned}
C_{f_1^1}(\bar{E}) &= ((a \cdot_a f_1(a)^{*a}) \cdot_a b) \cdot_b (f_1(a)^{*a} \cdot_a b + h_2(b))^{*b} \\
C_{h_2^1}(\bar{E}) &= b \cdot_b (f_1(a)^{*a} \cdot_a b + h_2(b))^{*b} \\
C_{g_3^1}(\bar{E}) &= (c \cdot_c g_3(c, a)^{*c}) \cdot_c (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b} \\
C_{g_3^2}(\bar{E}) &= (a \cdot_c g_3(c, a)^{*c}) \cdot_c (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b} \\
C_{f_4^1}(\bar{E}) &= ((a \cdot_a f_4(a)^{*a}) \cdot_a b) \cdot_b (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b} \\
C_{h_5^1}(\bar{E}) &= b \cdot_b (f_4(a)^{*a} \cdot_a b + h_5(b))^{*b}.
\end{aligned}$$

Applying $\sim_{\mathcal{F}}$ over the states of \mathcal{C}_E we get: $C_{f_1^1}(\bar{E}) \sim_{\mathcal{F}} C_{h_2^1}(\bar{E})$ then the two states are merged, $C_{f_4^1}(\bar{E}) \sim_{\mathcal{F}} C_{h_5^1}(\bar{E})$ so they are merged. The states $C_{g_3^1}(\bar{E})$ and $C_{g_3^2}(\bar{E})$ are not merged with anyone.

The number of states is $|Q| = 5$ and the number of transition rules is $|\Delta| = 15$.

The quotient automaton of this automaton by the equivalent relation $\sim_{\mathcal{F}}$ is given in Figure 6.

The quotient automaton of the automaton $\mathcal{C}_E / \sim_{\mathcal{F}}$ by the equivalent relation \sim_e is given in Figure 7. The number of states is $|Q| = 4$ and the number of transition rules is $|\Delta| = 14$.

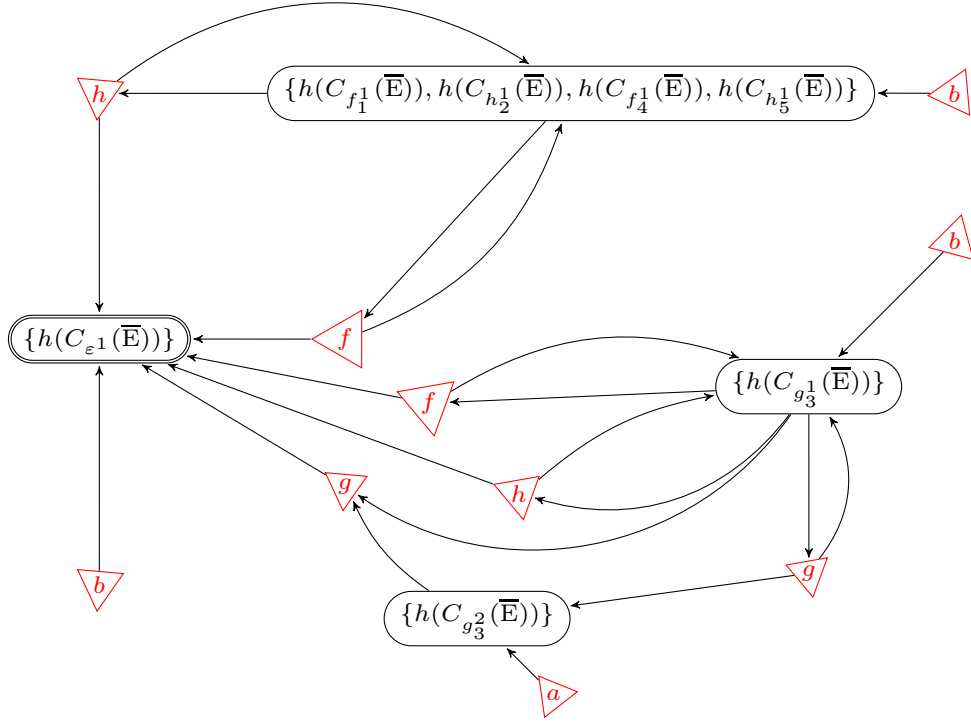


Fig. 7. The resulting automaton Automaton.

5 Conclusion

In this paper we define and recall different constructions of tree automata from a regular expression.

The different automata and their relations (quotient, isomorphism) defined in this paper are represented in Figure 8.

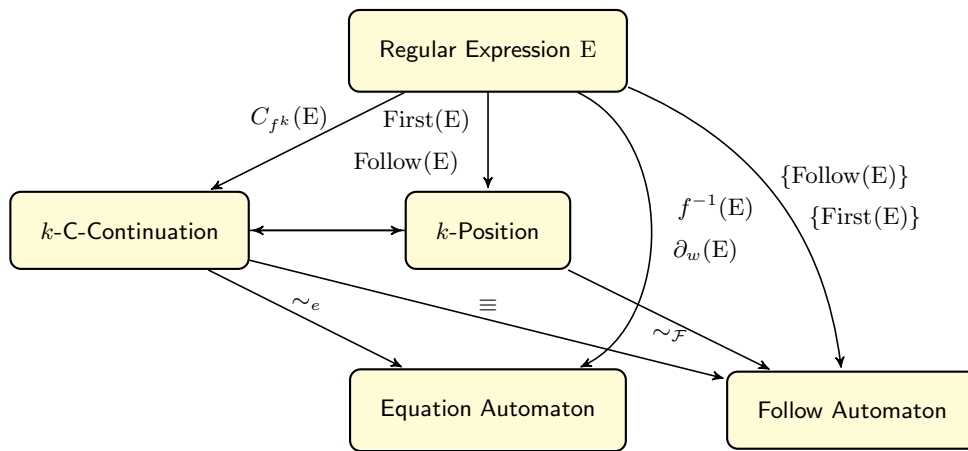


Fig. 8. Relation between Automata

Looking for reductions over the set of states, we applied the García *et al.* algorithm which allowed us to compute an automaton that assures that the size of the automata obtained is bounded above by the size of the smallest of the follow and the equation automata.

References

1. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. *Theor. Comput. Sci.* **155**(2) (1996) 291–319
2. Brzozowski, J.A.: Derivatives of regular expressions. *J. ACM* **11**(4) (1964) 481–494
3. Champarnaud, J.M., Ziadi, D.: From c-continuations to new quadratic algorithms for automaton synthesis. *IJAC* **11**(6) (2001) 707–736
4. Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. *Theor. Comput. Sci.* **289**(1) (2002) 137–163
5. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Loding, C., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata> (October 2007)
6. Cortes, C., Haffner, P., Mohri, M.: Rational kernels: Theory and algorithms. *Journal of Machine Learning Research* **5** (2004) 1035–1062
7. García, P., López, D., Ruiz, J., Alvarez, G.I.: From regular expressions to smaller nfes. *Theor. Comput. Sci.* **412**(41) (2011) 5802–5807
8. Glushkov, V.M.: The abstract theory of automata. *Russian Mathematical Surveys* **16** (1961) 1–53
9. Ilie, L., Yu, S.: Follow automata. *Inf. Comput.* **186**(1) (2003) 140–162
10. Khorsi, A., Ouardi, F., Ziadi, D.: Fast equation automaton computation. *J. Discrete Algorithms* **6**(3) (2008) 433–448
11. Kuske, D., Meinecke, I.: Construction of tree automata from regular expressions. *RAIRO - Theor. Inf. and Applic.* **45**(3) (2011) 347–370
12. Laugerotte, É., Sebt, N.O., Ziadi, D.: From regular tree expression to position tree automaton. In Dediu, A.H., Martín-Vide, C., Truthe, B., eds.: *LATA*. Volume 7810 of *Lecture Notes in Computer Science*, Springer (2013) 395–406
13. Mignot, L., Sebt, N.O., Ziadi, D.: An efficient algorithm for the equation tree automaton via the k-c-continuations. *Lecture Notes in Computer Science*, to appear (2014)
14. Mignot, L., Sebt, N.O., Ziadi, D.: An efficient algorithm for the equation tree automaton via the k-c-continuations. *CoRR* **abs/1401.5951** (2014)